Technical Report
CMU/SEI-92-TR-1
ESD-92-TR-1

2

Carnegie-Mellon University
Software Engineering Institute

Ada Validation Tests for Rate Monotonic Scheduling Algorithms

Keith A. Kohout
Kent Meyer
John B. Goodenough

February 1992

Technical Report
CMU/SEI-92-TR-1
ESD-92-TR-1
February 1992

Ada Validation Tests for Rate Monotonic Scheduling Algorithms

Keith A. Kohout
Naval Weapons Center

Kent Meyer
Telesoft, Inc.

John B. Goodenough
Rate Monotonic Analysis for Real-Time Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]

John S. Herman, Capt, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Table of Contents

List of Figures

Ada Validation Tests for Rate Monotonic Scheduling Algorithms

Abstract: This report presents a set of tests for checking whether an Ada runtime system properly supports certain rate monotonic scheduling algorithms, specifically, the *basic inheritance* and *priority ceiling* protocols. These tests are intended to be used by vendors and by users to validate implementations of these protocols. The report describes the tests and how they are to be used. The source code is available electronically.

1 Introduction

Rate monotonic scheduling (RMS) theory [Sha 90] allows a software designer to predict analytically whether a software system will meet its deadlines. The theory can also be used on developed software systems to help locate timing anomalies and performance bottlenecks. A major advantage of using RMS theory is that it allows systems to be built in which the software structure is not distorted by timing issues [Sha 90].

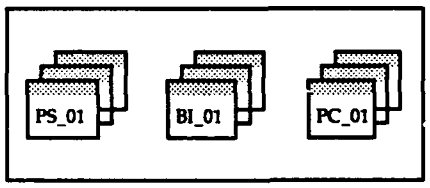
This report is designed to help runtime system developers and end users determine if a given Ada runtime system supports certain rate monotonic scheduling algorithms. Two related products are discussed: a set of tests that check for likely implementation errors in runtime system support for RMS algorithms and some tools that make it easier to use these tests.

Two types of tests are needed to check how well a runtime system supports a scheduling algorithm: *logical* tests that check whether the algorithm has been implemented correctly, and *performance* tests that measure the efficiency of an implementation. This report discusses only those logical tests needed to check the priority inheritance protocols [Sha 90] (basic inheritance and priority ceiling). The sporadic server algorithms [Sprunt 89] are not tested at this time; performance tests are under development.

This suite of tests assumes that Ada application programs use the *monitor task* paradigm to implement critical regions protected by a semaphore. A monitor task (called a *server* task in the remainder of this report) is a task that ensures mutually exclusive access to some resource through a critical region protected by a semaphore. In Ada, the way to represent such critical regions is with a task whose body is a single select statement enclosed in an endless loop. The task serves the role of the semaphore and the accept alternatives within the select statement serve as different critical regions. Non-server tasks are called *client* tasks. Such tasks may call server tasks. To model a nested critical region, a server task calls another server task.

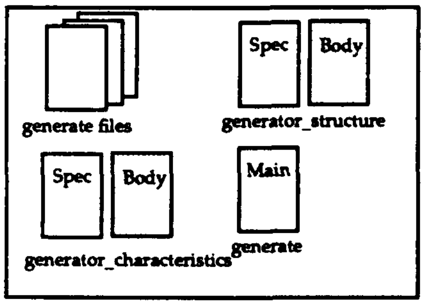
The *logical test system* described in this report contains several components (see Figure 1). The next section briefly describes the test objectives for each of the logical tests. (Detailed objectives are specified in Appendix A.) Section 3 describes a program generator for creating specified tests. Section 4 describes the test harness that supports the execution of specified tests. Section 5 describes a comparison tool for checking test results. The final section is a user's guide that describes the steps of compiling, executing, and evaluating the tests.

Test Specifications



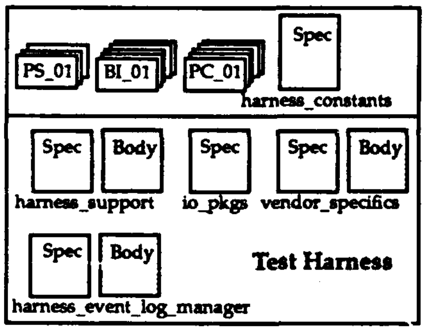
TEST SPECIFICATION
The logical test specifications are summarized beginning on page 3; detailed specifications are given in Appendix A.

Test Generator



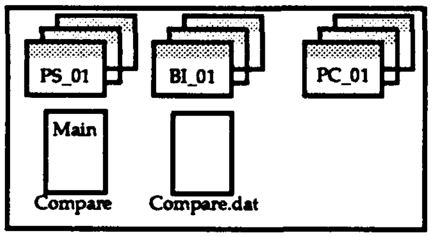
TEST GENERATOR
The logical test generator is described on page 9. This tool is written in Ada. It automatically generates test files. Two generator packages, generator_structure and test_characteristics, are used to specify the characteristics of the tests to be generated. The tests described in this document were generated with this tool. The generator is provided to allow further development or modification of test cases.

Logical Tests and Test Harness



LOGICAL TESTS and TEST HARNESS
The test harness that supports test case execution is discussed on page 11. The generator produces individual test case files and a harness_constants package. The generated test files "with" the routines provided in the test harness (event logging, I/O, vendor interface, etc). The test harness and the individual tests must be compiled together. Each test is linked into a separate load module that can be executed on the target machine. The comparator is then used to check the execution output of each test against the expected results. Some modification of the harness_constants and vendor_specifics packages may be required for a given implementation.

Comparator



COMPARATOR
The comparator is described on page 17. Like the generator, the comparator is written in Ada and runs on the host system. The comparator checks test execution output against the expected output (see Appendix B).

USER'S GUIDE
The user's guide, on page 21, presents the steps that should be taken to modify, compile, link, execute, and check the logical tests.

Figure 1 Logical Test System Components

2 Test Specifications

The tests described in this document check the logical behavior of a runtime system that supports the basic inheritance (BI) and priority ceiling protocols (PCP). Each test checks an important protocol behavior. Three categories of behavior are checked.

1. Priority service: executing tasks are preempted by tasks of higher priority.
2. Basic inheritance: a server is executed at either the priority of the task being served or at the priority of a caller whose call is unable to be accepted, whichever is higher. Moreover, blocked callers are serviced in order of priority, not in order of arrival.
3. Priority ceiling: in addition to basic inheritance, a server task is not allowed to accept a call unless its priority ceiling[1] is higher than the priority ceiling of any server that is executing a call on behalf of another client task. Since this rule introduces a new reason for not accepting entry calls, if a server's priority ceiling causes an entry call to be blocked, the server blocking the call inherits the blocked caller's priority.

The tests that follow are described in detail in Appendix A.

2.1 Priority Service (PS) Tests

Higher-priority clients, when they become eligible to run, must *preempt* lower-priority clients.

- **PS_01:** Check for preemption of lower-priority clients by higher-priority clients.

Low-priority clients are not allowed to preempt high-priority clients. In a priority-based scheduling system, only higher-priority clients are allowed to preempt. This is not true of a time-sliced scheduling system.

- **PS_02**: Check that low-priority client tasks are not executed in favor of high-priority clients (for two low-priority clients).
- **PS_03**: Check that low-priority client tasks are not executed in favor of high-priority clients (for four low-priority clients).

2.2 Basic Inheritance (BI) Tests

Medium-priority clients are prevented from executing (inheritance blocking) when a high-priority client is waiting to use a server that has been called by a low-priority client. This is because the server is executed with the waiting caller's priority.

- **BI_01**: Check for inheritance blocking. When a server is in rendezvous with a low-priority client, the server should inherit the priority of any (higher-priority) caller, thereby ensuring that medium-priority tasks do not execute.

1. The priority ceiling of a server is the highest priority of all client tasks that can call it directly or indirectly.

A server should *inherit the highest priority* of the clients that are waiting for the server. Moreover, queued clients should be serviced in priority order.

- **BI_02**: Check that a server executes at the highest priority of all waiting clients' priorities and that the server's priority is lowered after a high-priority client is served. Also, check that blocked clients are serviced in priority order, i.e., when more than one client is waiting for a server, the highest-priority waiting client should execute next.

The next tests check for *transitive inheritance,* namely, that a client's priority is transmitted through one server to subsequent servers.

- **BI_03**: Check for transitive inheritance of client's priority. A nested server should inherit the server's inherited priority, and hence, if a lower-priority task becomes ready to execute while a nested server call is being performed, the lower-priority task should not execute.

- **BI_04**: Check for transitive inheritance of a client's priority and verify that higher-priority clients can preempt the execution of a nested server call.

- **BI_05**: Check that if a high-priority client calls a server while it is executing a nested server call, no medium-priority tasks are allowed to execute (since the server inherits the priority of the waiting client).

Servers that are delayed (suspended while executing a delay statement) *do not block other tasks* from executing. This avoids unnecessary periods of idle time.

- **BI_06**: Check that a delayed server running on behalf of a high-priority client allows a lower-priority client to execute.

- **BI_07**: Check that a delayed server running on behalf of a high-priority client allows a lower-priority client to execute and call other servers. This test does not apply to implementations that support the priority ceiling protocol (see PC_07).

To minimize runtime system overhead, when a server has completed a rendezvous, it is expected to continue to execute until it is ready to accept another entry call instead of giving up control as soon as the rendezvous is completed.

- **BI_08**: Check that servers are either executing or are ready to accept an entry call.

Examples from CMU/SEI-89-TR-15 [Borger 89] that combine preemption, direct blocking, inheritance (push-through) blocking, simple inheritance, and transitive inheritance:

- **BI_09**: Check for preemption, blocking, inheritance, and proper priorities of tasks for BI (similar to Example #1 in CMU/SEI-89-TR-15). (PC_11 tests for behavior of the same task set when the priority ceiling protocol is supported.)

- **BI_10**: Check for preemption, blocking, inheritance, and proper priorities of tasks for BI (similar to Example #2 in CMU/SEI-89-TR-15). (PC_12 tests for behavior of the same task set when the priority ceiling protocol is supported.)

2.3 Priority Ceiling (PC) Tests

The priority ceiling algorithm, in its pure form, requires that a server execute at either the priority of its caller or at the priority of a blocked calling task, whichever is higher. The tests prepared in association with this paper check for this behavior. But there are two alternative forms of implementation that have essentially the same effect on worst-case schedulability and that can be easier or more efficient to implement. The simplest implementation is to execute each server at the highest possible priority. This means servers execute in a non-preemptible fashion. It also means that the worst-case blocking time for any task is the longest critical region (rendezvous) executed by any lower-priority server task. In cases where all critical regions are short, this approach has little effect on overall system schedulability. An additional advantage is that no caller is ever queued on a server as long as the server never suspends its execution; every server call can be accepted (on a uniprocessor architecture) because once the call is accepted, no other task can run to make any call; it is never necessary to make a context switch because a call is blocked. We call this form of implementation *the non-preemptible-server emulation method.*

The second implementation method (*the server-ceiling emulation method*) is to have each server execute at its ceiling priority. In this case, higher-priority tasks will be able to execute and will be able to call servers, thereby improving the average-case performance and in some cases, improving their worst-case performance over the emulation by preemption method. This method is only equivalent to the priority ceiling protocol if the server task does not suspend itself while a rendezvous, thereby allowing lower-priority tasks to run. The example in Figure A-23 (for PC_07) shows the kind of situation in which the server-ceiling emulation method could potentially lead to deadlock or chained blocking.

If an implementation supports one of the priority ceiling emulation methods, the sequence of task executions will, in general, be different. Specific differences are described in Appendix A.

In a pure implementation of the priority ceiling protocol, a high-priority call by a client to an available server can be blocked just because of the priority ceiling protocol rule; this form of blocking is called *ceiling* blocking.

- **PC_01**: Check that a high-priority task's call to an available server is blocked when the caller's priority does not exceed the priority ceiling of a server task that is in rendezvous with a lower-priority task.

Mutual deadlock is prevented by the priority ceiling protocol. The protocol prevents clients from deadlocking if they attempt to call the same set of servers in different orders.

- **PC_02**: Check for deadlock avoidance involving two clients.

If a high-priority client accesses two or more servers that are also accessed by lower-priority clients, the priority ceiling protocol ensures that the high-priority client will be blocked by at most one lower-priority client's rendezvous with a server. This is because medium-priority clients are not allowed to begin any server calls once a server with a high-priority ceiling is executing.

- **PC_03**: When lower-priority tasks access the same servers, check that a client is prevented from accessing its servers at most once, and that there are no nested server calls.
- **PC_04**: When there are nested server calls, check that a client is prevented from accessing its servers at most once when lower-priority tasks access the same servers.

Clients with priorities above a server's priority ceiling are *not blocked* from rendezvousing with other servers. The next tests check that servers do not execute above their priority ceiling.

- **PC_05**: Check that a client with a priority above a preempted server's ceiling can rendezvous with a ready server.
- **PC_06**: Check that a client with a priority above a preempted server's ceiling can rendezvous with a ready server. This test will fail if servers are executed at their ceiling priority.

Servers that are delayed *do not block ready tasks* from executing. However, if a lower-priority task, C2, attempts to call a server while a high-priority task is delayed in a server rendezvous, C2 must be prevented from calling the server, since C2's priority cannot exceed the ceiling of the executing server.

- **PC_07**: Check that a delayed server running on behalf of a high-priority client allows a lower-priority client to execute, but that calls to other servers by the lower-priority task will be blocked. (This test is the same as BI_07, but the expected results are different when the priority ceiling protocol is being supported.)

The next two tests provide somewhat more complex deadlock avoidance tests.

- **PC_08**: Check for deadlock avoidance (three clients).
- **PC_09**: Check for deadlock avoidance (three clients).

Periods of *idle time can exist* if no tasks are eligible to run. Servers that are delayed allow lower-priority ready tasks to execute as long as no server calls are made.

- **PC_10**: Check that delayed servers don't block lower-priority clients.

Examples from CMU/SEI-89-TR-23 [Borger 89] that combine preemption, direct blocking, inheritance (push-through) blocking, simple inheritance, transitive inheritance, and ceiling blocking:

- **PC_11**: Check for preemption, blocking, inheritance, and proper priorities of tasks for PCP (similar to Example #1 in CMU/SEI-89-TR-15). The output from this test will be different from the output for BI_09 even though the pattern of client and server calls is the same in both tests.

- **PC_12**: Check for preemption, blocking, inheritance, and proper priorities of tasks for PCP (similar to Example #2 in CMU/SEI-89-TR-15). As for PC_11, the output from this test will be different from the output for BI_10 even though the pattern of client and server calls is the same in both tests.

The next test checks that if a server is indirectly blocking one or more high-priority tasks, the server executes with the highest priority of the blocked tasks. This situation can arise when the priority ceiling of the server is higher than the priority of the calling tasks, and the calling tasks are calling other servers.

- **PC_13**: Check that a server is executed with the priority of tasks that are ceiling-blocked in attempting to call other servers. This test checks when there are two ceiling-blocked tasks and ensures that the highest-priority blocked task is executed first.

3 Test Generator

The test cases all have a very similar structure, so they can be generated automatically from a specification of test behavior. The specification format allows a user to describe test cases in terms of the test case diagrams shown in Appendix A. As new tests are developed or as tests change, a new diagram should be created that specifies the expected test behavior and the expected sequence of calls.

3.1 Requirements

The purpose of the generator is to produce a suite of test cases automatically from a description of the tests. The generator was developed to meet the following requirements.

1. Generate each test as a stand-alone program.
2. Create no dependent tasks (i.e., all tasks will be created at the library level). This provides compilers with the maximum opportunity to optimize server tasks and also reflects expected practice.
3. Allow for *task suspension* in a server. Task suspension represents a delay in processing. In a real system, a delay could be caused by I/O.
4. Comply with the recommended server coding style (which gives maximum opportunity for compiler optimization):
   - Set priority ceilings outside of the server tasks.
   - Have the generator compute the priority ceiling of each server task. This keeps the user from having to specify the server's priority ceiling.
   - Use the terminate alternative in servers rather than a STOP entry call to shut down server tasks.
   - Don't use an exception handler in the body of a server.
   - Don't give server tasks a priority using pragma PRIORITY. This allows the priority inheritance and the priority ceiling protocols to be supported by a validated Ada compiler [Sha 90]. (Servers can be given a priority by some other mechanism, however, such as a runtime system call.)
5. Isolate possible test harness dependencies (e.g. TEXT_IO, delay_until). This separates the possible changes that users may have to make.
6. In a client's exception handler, spell out "Unexpected exception in Client_N Task". This gives more information to the user.
7. Rather than placing all clients and servers in one package, place *each* client and server in its *own* package. This better mirrors an actual code structure that would be used in practice.
8. Allow the generator to accept an arbitrary number of parameters that specify the structure of the tasks for the given logical test. This allows for maximum flexibility.

3.2 Customization

The areas of customization for the generator have been localized to two packages. Each of these packages will need to be examined. The generator may be customized with respect to task names, priorities, etc.:

- Generator_Structure. The following can be found in the Generator_Structure package (`gensts.ada`).

```
---------- User Customization Section Begins ----------
------
-- Customization for Constraints on Generator
------
Max_Number_Of_Tests: constant Positive := 30;
Max_Task_Events     : constant Positive := 5;
Max_Server_Entries  : constant Positive := 5;

type Task_Names is (C1, C2, C3, C4, C5, S1, S2, S3, S4, S5);
type Entry_Name_Types is (E1, E2, E3, E4, E5);
------
-- Customization for Build_Harness_Constants
------
type Byte is range 0 .. 255;                    -- Type Byte Needed
for Byte'Size use 8;

Event_Logger_Priority  : constant Byte := 60;    -- Highest Priority
Server_Priority        : constant Byte := 32;    -- Server Priority
Main_Priority          : constant Byte := 1;     -- Lowest Priority
Test_Done_Priority     : constant Byte := 59;    -- Next Highest Priority
Settling_Time          : constant Float := 1.0;  -- Startup time
Interactive_Trace      : constant Boolean := False; -- Debug Off

-- Priorities P1-P10 should be higher than the Server_Priority.
type Priority_Type is (P1, -- Low Priority
                       P2, P3, P4, P5, P6, P7, P8, P9,
                       P10); -- High Priority
for Priority_Type use (P1 => 33, P2 => 34,
                       P3 => 37,   -- Actual values can be noncontiguous
                       P4 => 38, P5 => 41, P6 => 43, P7 => 46, P8 => 47, P9 => 48, P10 => 49);
for Priority_Type'Size use 8;
---------- User Customization Section Ends ----------
```

- Test_Characteristics (`tstchb.ada`). Customization is required only if the user needs to change the suite of logical tests. This involves modifying the parameters that describe the test suite to be generated. Modifying the Test_Characteristics package is easier than creating this package anew. The specification for test BI_05 is given in Appendix C on page 87; the diagram describing this test case is found in Appendix A on page 27.

4 Test Harness and Test Coding Style

The test harness is a package of subprograms that support the execution of test cases. In particular, various setup procedures are provided together with procedures for consuming time and for logging events that occur during execution. An automated examination of the event log determines whether a given test has been passed.

Clients and servers comprising a test are assumed to be coded in certain patterns of calls to the test harness. The next section discusses the coding style required for servers. Section 4.2 discusses various services provided by the test harness.

4.1 Server Coding Style

The test case generator automatically produces server tasks written in a style that allows servers to be implemented efficiently as critical regions protected by a semaphore [Borger 89]. These restrictions are:

1. All accept statements must be contained in a single select statement that is the only statement in the body of an endless loop.
2. A server task is not assigned a priority using the pragma PRIORITY. Since Ada's rules do not specify any particular priority or scheduling policy for tasks that do not have a defined priority, the runtime system can allow these servers to execute in accordance with the basic inheritance and priority ceiling protocol rules [Sha 90].

The sample Ada code segment shown in Figure 2 illustrates the server coding style used in the tests. The example server task (S1) has a single select statement with two accept alternatives (E1 and E2) and a terminate alternative. The terminate alternative will be selected when the test is finished.

4.2 Test Harness Services

The event logger captures the activities of a test case execution. As the test harness executes, calls to the logger capture the identity of the active task (a client or a server), the identity of the calling client task (when an event is logged within a server), the current time, and the nature of the event. A printout of logged events is usually provided after the test case has finished, but printouts can be provided interactively. Reporting events as they occur can cause unintended test behavior because of the time taken to print out the information, but interactive reporting can be useful when debugging a test case.

```
task body S1_Task is
  Exec_Time_1 : constant Duration := 1.0;
  Exec_Time_2 : constant Duration := 3.0;
begin
  --------
  -- Get my id
  --------
  accept Get_Id (Task_Id : out Vendor_Specifics.Task_Id) do
    Task_Id := Vendor_Specifics.Get_Task;
  end Get_Id;
  --------
  -- Endless loop servicing clients
  --------
  loop
    select
      accept E1 (Caller_Id : in Task_Id_Type) do
        --------
        -- Server Execution
        --------
        Log_Event(S2, Caller_Id, Server_Exec_Begins);
        Harness_Support.Spend_Time(Exec_Time_1);
        Log_Event(S2, Caller_Id, Server_Exec_Ends);
      end E1;
    or
      accept E2 (Caller_Id : in Task_Id_Type) do
        --------
        -- Server Execution
        --------
        Log_Event(S2, Caller_Id, Server_Exec_Begins);
        Harness_Support.Spend_Time(Exec_Time_2);
        Log_Event(S2, Caller_Id, Server_Exec_Ends);
      end E2;
    or
      terminate; -- when test is finished
    end select;
  end loop;
end S1_Task;
```

Figure 2 Server Coding Style

4.2.1 Events to be Logged

The events that can be logged are defined by the type `Task_Exec_Phase`. Currently the events that can be logged are the following:

- Client_Exec_*Begins* – Beginning client execution. Followed by a call to Spend_Time.
- Client_Exec_*Ends* – Ending client execution.
- Task_Suspension_*Begins* – Beginning task suspension. Followed by a call to Suspend_Task.
- Task_Suspension_*Ends* – Ending task suspension.
- Server_Call_*Begins* – Beginning a call to a server. Followed by the actual server call. Both clients and servers can call other servers.
- Server_Call_*Ends* – Ending a server call.

- Server_Exec_*Begins* – Beginning server execution. Followed by a call to Spend_Time.
- Server_Exec_*Ends* – Ending server execution.

Note: All of the *Begins* and *Ends* above are paired to represent actions that can be repeated more than once for a given client or server. A client typically performs more than one action.

4.2.2 Client/Server Event Patterns

A *client* may contain any combination of the following patterns of event logging calls and calls to harness support procedures. The Log_Event call takes three parameters: the first is the identifier of the task containing the call; the second is used in servers and identifies the calling client or server task; the third identifies the event that is to be logged.

- Execution time spent in client (C1).

```
Log_Event(C1, None, Client_Exec_Begins);
Harness_Support.Spend_Time(Exec_Time_1);
Log_Event(C1, None, Client_Exec_Ends);
```

- Call by a client (C1) to a server (S1).

```
Log_Event(C1, S1, Server_Call_Begins);
Server_S1_Package.S1_Task.E1(C1);
Log_Event(C1, S1, Server_Call_Ends);
```

A *server* may contain any combination of the following patterns:

- Execution time consumed by a server (S1) on behalf of a caller.

```
Log_Event(S1, Caller_Id, Server_Exec_Begins);
Harness_Support.Spend_Time(Exec_Time_1);
Log_Event(S1, Caller_Id, Server_Exec_Ends);
```

- Call by a server (S1) to another server (S2).

```
Log_Event(S1, S2, Server_Call_Begins);
Server_S2_Package.S2_Task.E1(S1);
Log_Event(S1, S2, Server_Call_Ends);
```

- Suspension of a server task (S1) that is executing on behalf of some caller.

```
Log_Event(S1, Caller_Id, Task_Suspension_Begins);
Harness_Support.Suspend_Task(Suspension_Time_1);
Log_Event(S1, Caller_Id, Task_Suspension_Ends);
```

4.2.3 Event Timeline Example

The output from the test harness is a log of events that should be compared with expected test results (see Appendix B). These events indicate the order of execution of clients and servers during the execution of a test case. The expected sequence of events is derived from the test case diagrams given in Appendix A. The example given in Figure 3 shows the expected output for test case "BI_05" (page 36).

```
[Task: C3 Begins Execution                              at t = 1]
[Task: C3 Ends Execution                                at t = 2]
[Task: C3 Calls server:                        S1       at t = 2]
[Task: S1 Begins Execution on behalf of:       C3       at t = 2]
[Task: S1 Ends Execution on behalf of:         C3       at t = 4]
[Task: S1 Calls server:                        S2       at t = 4]
[Task: S2 Begins Execution on behalf of:       S1       at t = 4]
[Task: C2 Begins Execution                              at t = 5]
[Task: C1 Begins Execution                              at t = 6]
[Task: C1 Ends Execution                                at t = 7]
[Task: C1 Calls server:                        S1       at t = 7]
[Task: S2 Ends Execution on behalf of:         S1       at t = 8]
[Task: S1 Begins Execution on behalf of:       C3       at t = 8]
[Task: S1 Ends Execution on behalf of:         C3       at t = 10]
[Task: S1 Begins Execution on behalf of:       C1       at t = 10]
[Task: S1 Ends Execution on behalf of:         C1       at t = 11]
[Task: C1 Begins Execution                              at t = 11]
[Task: C1 Ends Execution                                at t = 12]
[Task: C2 Ends Execution                                at t = 13]
[Task: C3 Begins Execution                              at t = 13]
[Task: C3 Ends Execution                                at t = 14]
******************* Test Complete *******************
```

Figure 3 Example Timeline

4.2.4 Additional Software Support

The test harness contains additional software that is used to calibrate a test case, start and stop the test case, activate the event logger, and print the timeline of events captured by the event logger. The command sequence for the main procedure for running tests such as BI_05 is shown below:

```
Harness_Support.Calibrate_Spend_Time;                  -- Calibration
Harness_Event_Log_Manager.Initialize(Start_Time);      -- Initialize Logger
bi_05_Test_Harness.Start_Run(Start_Time);              -- Start Clients
bi_05_Test_Harness.Test_Done.Complete;                 -- Wait for Clients
Harness_Event_Log_Manager.Print_Time_Lines;            -- Print Time Line
Harness_Event_Log_Manager.Quit;                        -- Stop Logger
```

The complete code for test case BI_05 is found in Appendix D.

4.3 Customization

The test-harness may be customized in the following areas:

- Vendor_Specifics. The following can be found in the Vendor_Specifics package. It should not be necessary to change the Vendor_Specifics package specification except possibly for its types and subtypes. The body of this package will need to be modified to support an individual vendor. The following code shows the TeleSoft 3.23 TeleAda-Exec vendor-specific package, `vendor.tel`:

```
with System;
with Calendar;
with Ada_Tasking_Extensions;
package Vendor_Specifics is
   subtype Task_Id is Ada_Tasking_Extensions.Task_Id;

   function Get_Task return Task_Id renames
       Ada_Tasking_Extensions.Current_Task;

   procedure Change_Priority (Of_Task : Ada_Tasking_Extensions.Task_Id;
                              New_Priority : System.Priority);

   procedure Set_Priority_Ceiling (Of_Task : Ada_Tasking_Extensions.Task_Id;
                                   New_Priority : System.Priority);

   function Get_Priority_Ceiling (Of_Task : Ada_Tasking_Extensions.Task_Id)
       return System.Priority renames Ada_Tasking_Extensions.Get_Priority;

   procedure Delay_Until(Wake_Up_Time: in Calendar.Time);
end Vendor_Specifics;

---------
-- Package Body
---------
with Ada_Tasking_Extensions;
package body Vendor_Specifics is
   procedure Change_Priority (Of_Task : Ada_Tasking_Extensions.Task_Id;
                              New_Priority : System.Priority) is
     Priority : Ada_Tasking_Extensions.Task_Priority;
   begin
     Priority := Ada_Tasking_Extensions.Change_Priority (Of_Task, New_Priority);
   end Change_Priority;

   procedure Set_Priority_Ceiling (Of_Task : Ada_Tasking_Extensions.Task_Id;
                                   New_Priority : System.Priority) is
     Priority : Ada_Tasking_Extensions.Task_Priority;
   begin
     Priority := Ada_Tasking_Extensions.Change_Priority (Of_Task, New_Priority);
   end Set_Priority_Ceiling;

   procedure Delay_Until(Wake_Up_Time: in Calendar.Time) is
     Now : Calendar.Day_Duration := Calendar.Seconds(Calendar.Clock);
   begin
     delay(Calendar.Seconds(Wake_Up_Time) - Now);
   end Delay_Until;
end Vendor_Specifics;
```

- Modifications to Harness_Constants (`hconst.ada`) should not be needed if the generator creates this package.
- Modifications to IO_Pkgs package (`iopkgs.ada`) should be needed only if this package does not compile. This would indicate different I/O compiler support.

5 Comparator

The comparator compares the actual event sequences of a test case execution with the expected sequence of events. For a test to pass, the actual and expected sequences of events must match line for line, excluding the time stamp. The time stamp is not considered because the overhead of the event logger and other execution times that are not accounted for will cause the time to vary slightly. At the completion of a test, the last event's time stamp is checked for a drift of no more than ±5%. If the drift was more than ±5%, a warning is given to check the execution times.

The input to the comparator is the data in the file `compare.dat`. Entries in this file are a succession of lines having the following format:

- the keyword Compare
- the location of the expected output
- the location of the actual output
- a descriptive comment

and a blank line. Copying and modifying this file is easier than creating it anew. An example file is shown in Figure 4.

```
****** Data File for Comparator ********
Compare
../Expected/ex_01.out
../Actual/ex_01.out
ex_01

Compare
../Expected/ex_02.out
../Actual/ex_02.out
ex_02

Compare
../Expected/ex_03.out
../Actual/ex_03.out
ex_03

Compare
../Expected/ex_04.out
../Actual/ex_04.out
ex_04

Compare
../Expected/ex_05.Oout
../Actual/ex_05.out
ex_05

Compare
../Expected/ex_06.out
../Actual/ex_06.out
ex_06
```

Figure 4 Comparator Data File Example

5.1 Comparator Output and Error Messages

Figure 5 shows example output from the comparator.

For ex_01, the test case passed. This means that the expected and the actual events matched and the completion times differed by less than ±5%.

For ex_02, the test case passed with a warning. This means that the expected and actual events matched but the completion times differed by more than ±5%. The completion times for the expected and actual results are printed.

For ex_03, the test case failed because the expected and actual events did not match. The place of the mismatch is shown by looking at the two lines (expected event and the actual event).

```
Test Suite

Test: ex_01 => Passed


Test: ex_02 => Passed {check times}
 ** Expected_Time = 10, Actual_Time = 9.100 **


Test: ex_03 => FAILED
[Task: S1 Begins Execution on behalf of: C1 at t = 7]
[Task: C1 Ends Call with server: S1 at t =


Test: ex_04 => FAILED ** ERROR -- Data_Error unexpected **
[Task: S2 Ends Suspension on behalf of: C2 at t = 6]
[Task: C1 Ends Call with server: S1 at t =


Test: ex_05 => FAILED ** ERROR -- Name_Error unexpected **
 ** File = ../test-suite/Expected/ex_05.Oout, could not be Opened **

Test: ex_06 => FAILED ** ERROR -- End-Of-File unexpected **
[Task: C3 Ends Execution at t = 8]
[Task: C3 Begins Execution at t =
```

Figure 5 Comparator Output Example

For ex_04, the test case failed with an unexpected error indication. The expected and actual events did not match because Data_Error was raised. The expected event and the actual event are printed.

For ex_05, the results file could not be opened. This file name is printed following the error message.

For ex_06, the end of file was encountered unexpectedly. The events before the unexpected end-of-file occurred are printed.

5.2 Customization

The comparator should not require customization. But the file `compare.dat` should change as the suite of tests changes.

6 User's Guide

6.1 Directory Release Structure

The directory release structure for the software that accompanies this report is shown in Figure 6.

```
<Release Directory>
    document
        Postscript of this document
        events
            ps_01.out (thru) ps_03.out
            bi_01.out (thru) bi_10.out
            pc_01.out (thru) pc_13.out
    tests
        harness
            evlogb.ada
            evlogs.ada
            hsuppb.ada
            hsupps.ada
            iopkgs.ada
            vendor.{aly | tel | trt | vdx}
        logicals
            ps_01.ada (thru) ps_03.ada
            bi_01.ada (thru) bi_10.ada
            pc_01.ada (thru) pc_13.ada
    tools
        compare
            compare.ada
            compare.dat
        generate
            genera.ada
            gensep.ada
            genstb.ada
            gensts.ada
            tstchb.ada
            tstchs.ada
```

Figure 6 Directory Release Structure

6.2 Testing Process

This section steps through the use of the test case generator, the running of individual test cases, and the running of the comparator. The test case generator allows rapid development of a test case. The generator takes an Ada description of the test cases to be generated and produces appropriate packages. These packages and the test harness are then compiled together to produce an executable test program. The results of the test execution are then compared with the expected results. The test case generator and the comparator were designed to be run on the host machine, while the test harness was designed to be run on the target machine (which could be the host).

6.2.1 Test Generator Usage

Step 1: Generate the tests. The first step in the testing process is to produce the test files. The tests described in this report have already been created and are supplied with the software release. If you do not intend to make any major modifications or enhancements to these tests, you will not need to use the test generator tool, and you can skip to Step 2.

The input to the generator is a file containing the package Test_Characteristics (`tstchb.ada`). This package contains a description of the test cases to be generated (see Appendix C for an example). This file can be modified to produce new or different tests. The steps to follow in compiling the generator tool and specifying the tests it will produce are as follows:

1. Locate the directory `<Release_Directory>/tools/generate`. This directory contains all of the generator source files.
2. The source files `tstchb.ada` and `gensts.ada` should be modified according to the customization suggestions contained in Section 3.2 on page 10.
3. The generator should be compiled using a HOST Ada compilation system (the executable will run on the development machine). The main unit is `generate`, and the compilation order for these packages is shown in Figure 7.
4. Once the generator has been compiled and linked, running it will produce the `harness_constants` package and the individual test files (see Appendix D on page 91 for an example of a test).

6.2.2 Test Harness Usage

Step 2: Compile the tests and the test harness. The test harness is compiled along with the generated logical test files to produce the executable load modules that represent each test case. The tests may be ones created using the generator, or they may be the versions provided in the `tests/logicals` directory. To compile the test harness, follow these steps:

File Name	Unit(s)
gensts.ada	GENERATOR_STRUCTURE
genstb.ada	GENERATOR_STRUCTURE *<body>*
tstchs.ada	TEST_CHARACTERISTICS
tstchb.ada	TEST_CHARACTERISTICS *<body>*
genera.ada	GENERATE <body>
gensep.ada	GENERATE.BUILD_BODIES *<body>*
	GENERATE.BUILD_HARNESS_CONSTANTS *<body>*
	GENERATE.BUILD_MAIN *<body>*
	GENERATE.BUILD_SPECS *<body>*
	GENERATE.CALC_CEILINGS *<body>*
	GENERATE.MAKE_STRING *<body>*

Figure 7 Compilation Order for Generator Tool

1. Modify the test harness files located in `tests/harness` (*only* if they cannot otherwise be compiled).

   `iopkgs.ada`

   `hsuppb.ada`

   `vendor.tel` (or other vendor-specific package)

2. Modify the `hconst.ada` package located in `<Release_Directory>/tests/logicals` if it does not compile.

3. Compile the test harness and the `harness_constants` package using the TARGET Ada compiler. The compilation order is shown in Figure 8.

4. Compile and link the individual logical tests. All the unit names for the test cases are unique, so one library can contain the entire set of tests. For the test cases presented in this report, the file name reflects the main unit name, so an example compile and link would be the following:

   ```
   compile bi_05.ada

   link bi_05
   ```

5. Run the executable load module for each test, directing or capturing the output to a file, e.g., for a target running Unix:

   ```
   bi_05 > bi_05.out
   ```

File Name	Unit(s)
`hconst.ada`	HARNESS_CONSTANTS
`iopkgs.ada`	IO_PKGS
`vendor.???`	VENDOR_SPECIFICS
	VENDOR_SPECIFICS *<body>*
`evlogs.ada`	HARNESS_EVENT_LOG_MANAGER
`evlogb.ada`	HARNESS_EVENT_LOG_MANAGER *<body>*
`hsupps.ada`	HARNESS_SUPPORT
`hsuppb.ada`	HARNESS_SUPPORT *<body>*

Note: the above **???** in vendor.??? is specific to the tested compiler (tel, vdx, aly, ..)

Figure 8 Compilation Order for the Test Harness Files

6.2.3 Comparator Usage

Step 3: Check the test output. The input to the comparator is the file `compare.dat`. The format to be used for this file is described on page 17. The steps for creating the comparator program are as follows:

1. Compile `<Release_Directory>/tools/compare/compare.ada` and link the main unit `compare`, using a HOST compilation system.
2. Modify `compare.dat` (to reflect the test case output to be checked).
3. Run `compare`.

References

[Altman 87] Altman, Neal. *Factors Causing Unexpected Variations in Ada Benchmarks* (CMU/SEI-87-TR-22, DTIC: ADA187231). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, October 1987.

[Borger 89] Borger, Mark W.; & Rajkumar, Ragunathan. *Implementing Priority Inheritance Algorithms in an Ada Runtime System* (CMU/SEI-89-TR-15, DTIC: ADA209607). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April 1989.

[Sha 90] Sha, L.; & Goodenough, J. B. "Real-Time Scheduling Theory and Ada." *IEEE Computer 23*, 4 (April 1990):53-62.

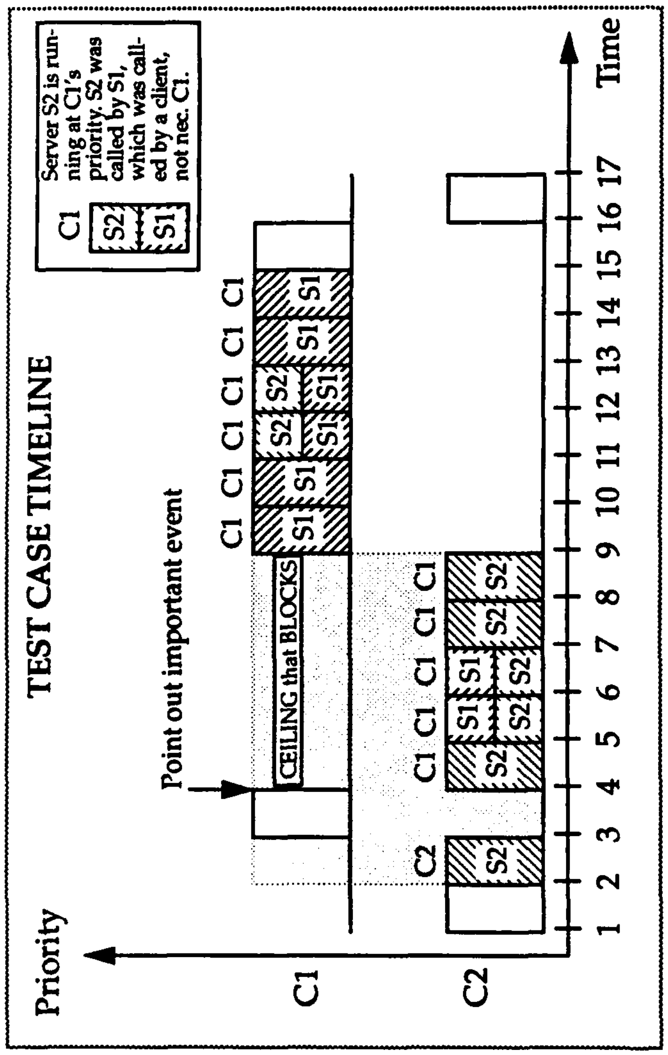
[Sprunt 89] Sprunt, B.; & Sha, L. *Scheduling Sporadic and Aperiodic Events in a Hard Real-Time System* (CMU/SEI-89-TR-11, DTIC: ADA211344). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April 1989.

Appendix A Logical Test Suite Diagrams

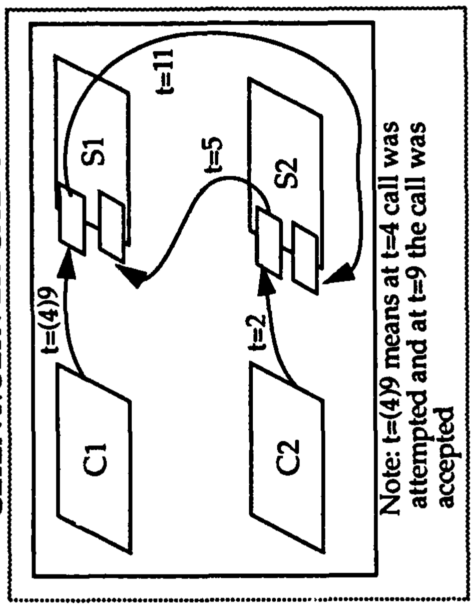
This Appendix contains diagrams describing the logical tests. The diagrams show the client and server tasks, their calling relationships, and the expected sequence of events. An explanatory diagram showing the graphic conventions appears on the next page. The rest of the Appendix contains diagrams for the tests described briefly in Section 2.

For the priority ceiling protocol tests, each diagram is annotated with a comment explaining any difference in test behavior under the two emulation methods: the method in which servers are executed non-preemptively (*the non-preemptible-server emulation method*), and the method in which servers are executed at their ceiling priority (*the server-ceiling emulation method*). If the difference in test behavior is the same under both emulation methods, we say so explicitly. For example, in Figure A-17, the change in test behavior is the same under both emulation methods, so we use the phrase "under either emulation method". In Figure A-21, the test behavior only changes when the non-preemptible-server emulation method is used, so the comment only mentions the change in this case.

TEST CASE TIMELINE



CLIENT/SERVER CALLS


C1
C2
S1
S2
t=(4)9
t=2
t=5
t=11
Note: t=(4)9 means at t=4 call was attempted and at t=9 the call was accepted

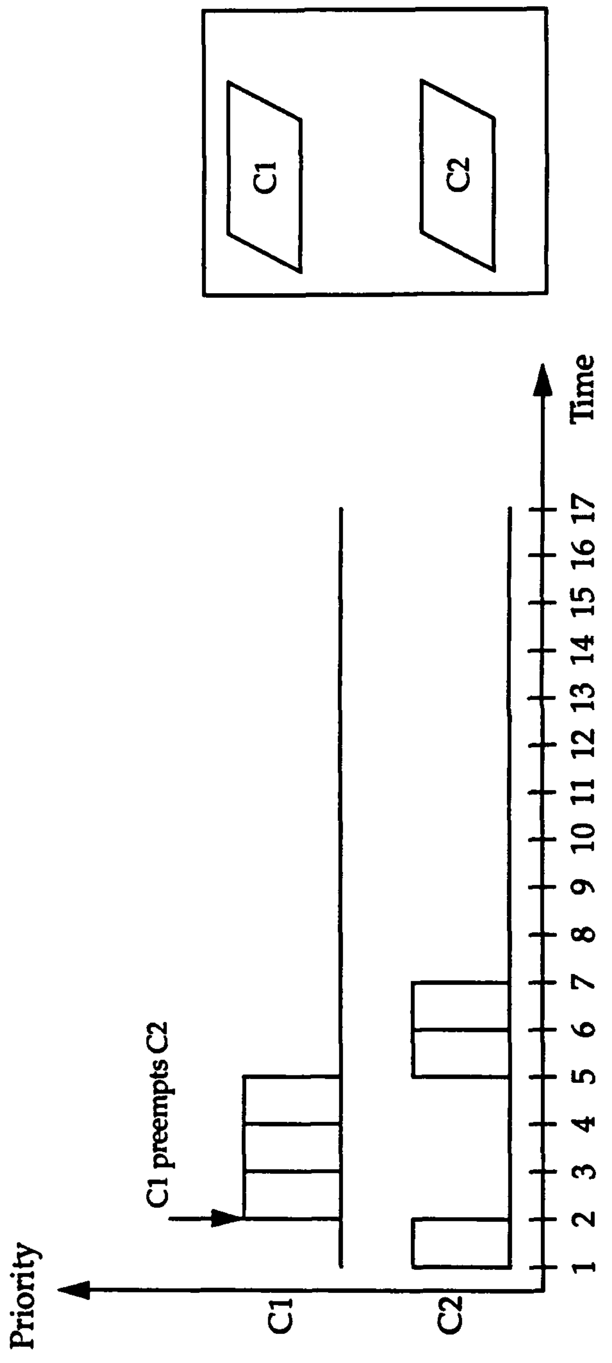
REASON for TEST CASE

Check an algorithm characteristic

EVENTS of TIMELINE

t=1 : C2 begins to execute
t=2 : C2 rendezvous with S2
t=3 : C1 preempts S2
t=4 : C1 attempts to call S1 [CEILING BLOCKED]
S2 resumes on behalf of C2
t=5 : S2 rendezvous with S1
t=7 : S1 completes and S2 resumes
t=9 : S2 completes
C1 preempts C2 and rendezvous with S1
t=11 : S1 rendezvous with S2
t=13 : S2 completes and S1 resumes
t=15 : S1 completes and C1 resumes
t=16 : C1 completes and C2 resumes
t=17 : C2 completes

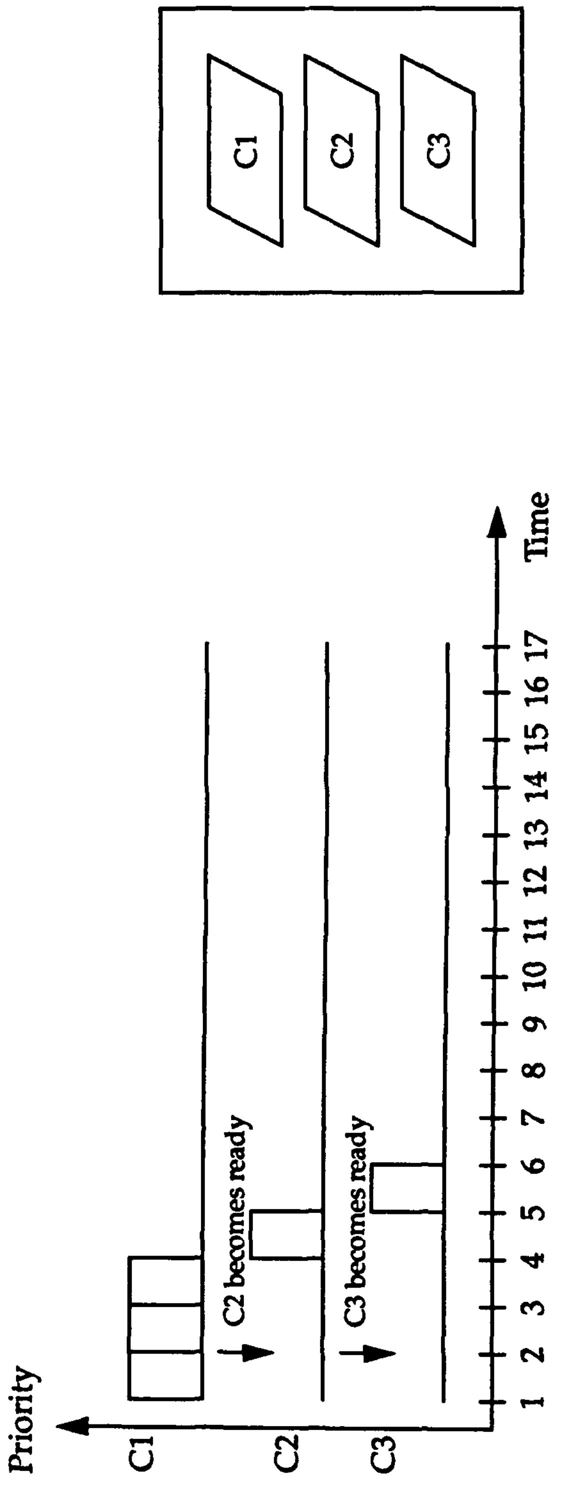
Figure A-1 Example Test Case



Check that high-priority clients preempt lower-priority clients

t=1 : C2 begins to execute
t=2 : C1 preempts C2
t=5 : C1 completes and C2 resumes
t=7 : C2 completes

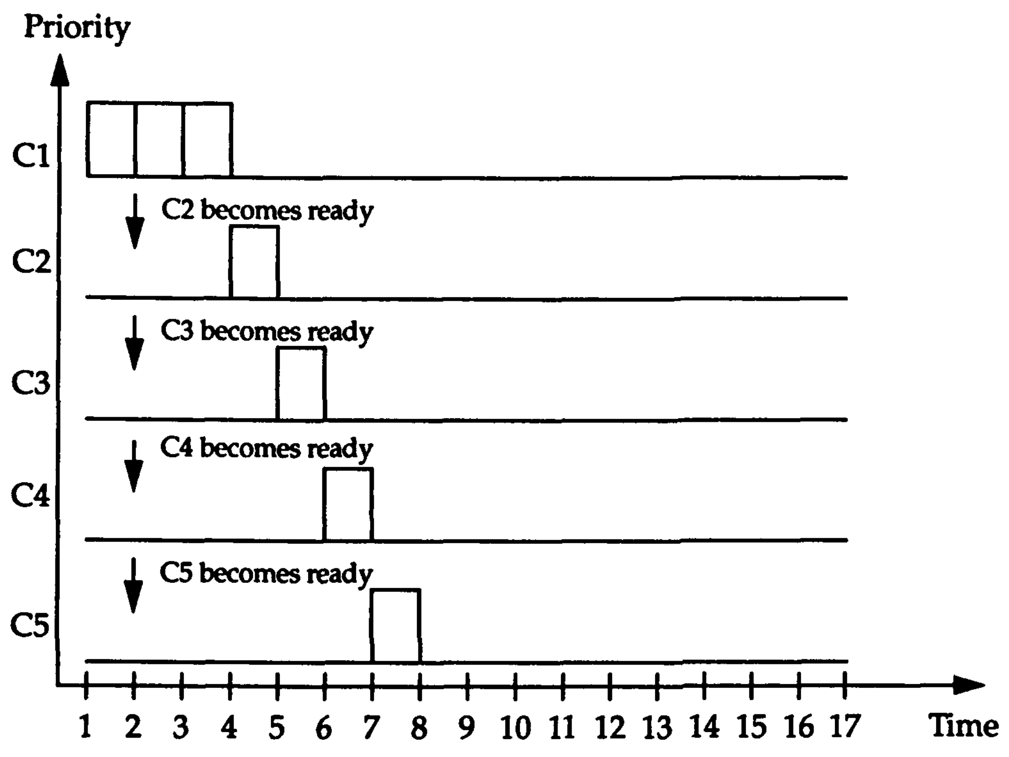
Figure A-2 PS_01: High-Priority Clients Preempt

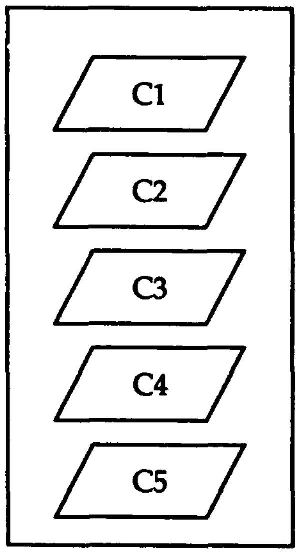


Check that *preemption* does NOT occur by lower-priority clients

t=1 : C1 begins to execute
t=2 : C2 & C3 become ready but do NOT preempt
t=4 : C1 completes and C2 begins to execute
t=5 : C2 completes and C3 begins to execute
t=6 : C3 completes

Figure A-3 PS_02: Low-Priority Clients (2) Don't Preempt

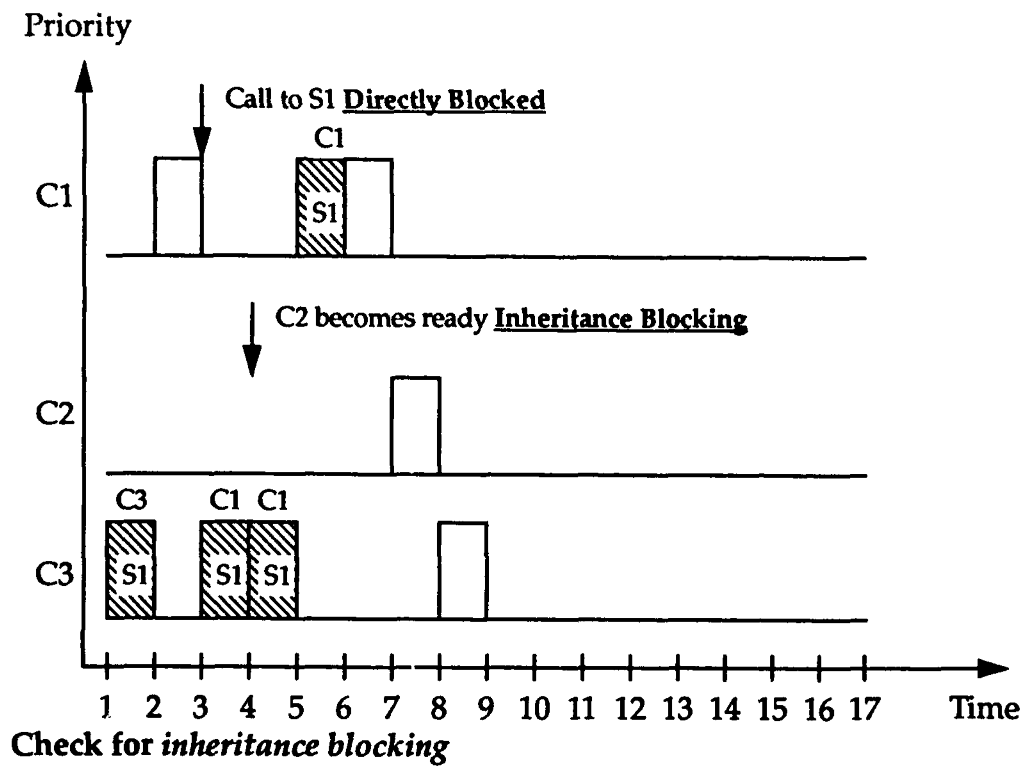


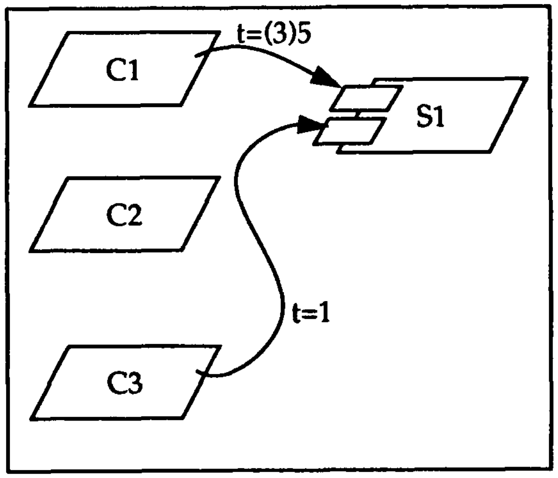


Check that *preemption* does NOT occur by lower-priority clients

t=1 : C1 begins to execute
t=2 : C2, C3, C4, C5 become ready but do NOT preempt
t=4 : C1 completes and C2 begins to execute
t=5 : C2 completes and C3 begins to execute
t=6 : C3 completes and C4 begins to execute
t=7 : C4 completes and C5 begins to execute
t=8 : C5 completes

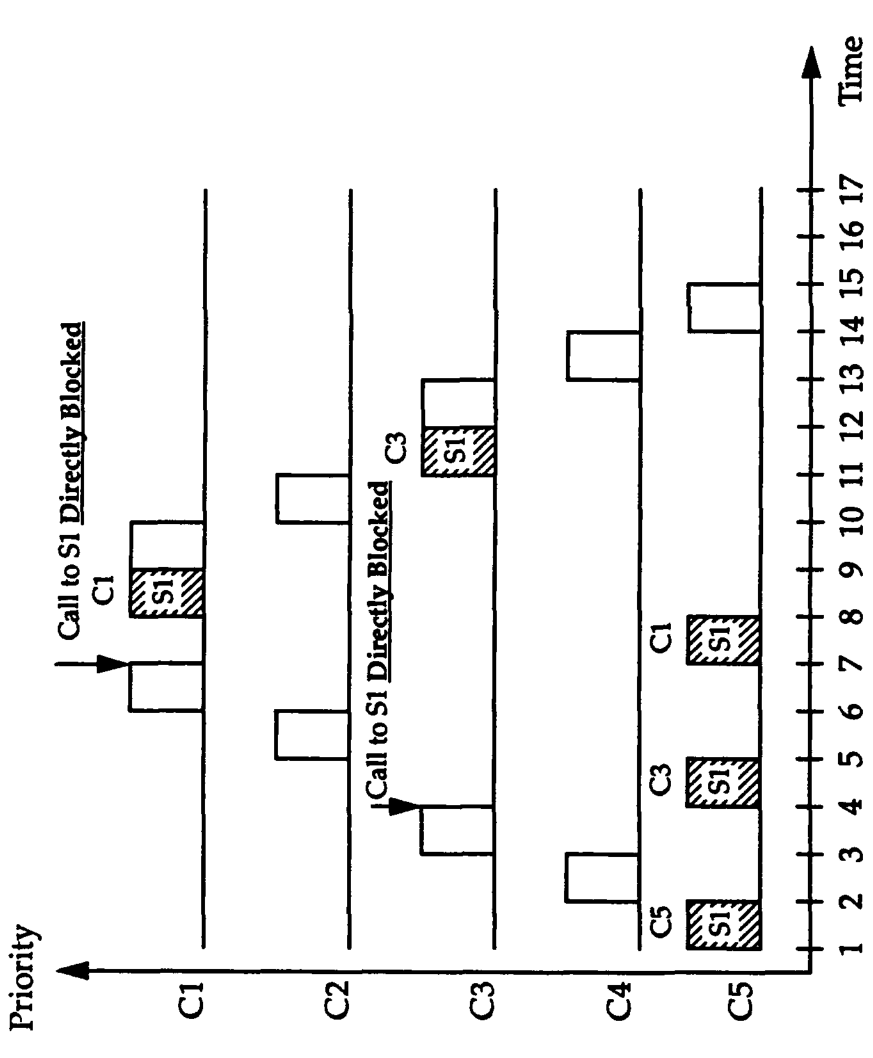
Figure A-4 PS_03: Low-Priority Clients (4) Don't Preempt



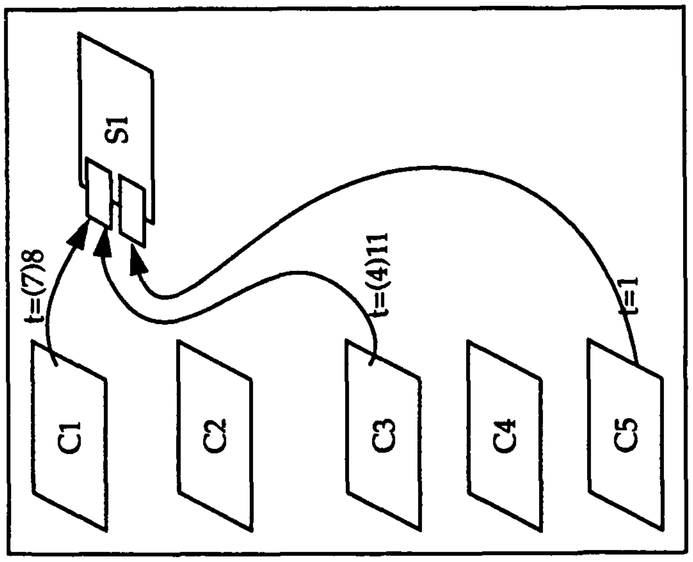


t=1	:	C3 rendezvous with S1
t=2	:	C1 preempts S1
t=3	:	C1 attempts to call S1 [DIRECTLY BLOCKED]
		S1 resumes on behalf of C3
t=4	:	C2 becomes ready [INHERITANCE BLOCKING]
		S1 continues on behalf of C3
t=5	:	S1 completes
		C1 preempts C3 and rendezvous with S1
t=6	:	S1 completes and C1 resumes
t=7	:	C1 completes
		C2 preempts C3
t=8	:	C2 completes and C3 resumes
t=9	:	C3 completes

Figure A-5 BI_01: Check for Inheritance Blocking



Check that *server inherits and disinherits* highest waiting client's priority



t=1 : C5 rendezvous with S1
t=2 : C4 preempts S1
t=3 : C3 preempts C4
t=4 : C3 attempts to call S1 [DIRECTLY BLOCKED]
S1 resumes on behalf of C5
t=5 : C2 preempts S1
t=6 : C1 preempts C2
t=7 : C1 attempts to call S1 [DIRECTLY BLOCKED]
S1 resumes on behalf of C5
t=8 : S1 completes and C1 resumes
C1 rendezvous with S1
t=9 : S1 completes and C1 resumes
t=10: C1 completes and C2 resumes
t=11: C2 completes
C3 resumes and rendezvous with S1
t=12: S1 completes and C3 resumes
t=13: C3 completes and C4 resumes
t=14: C4 completes and C5 resumes
t=15: C5 completes

Figure A-6 BI_02: Server Priority Is Raised In Stages

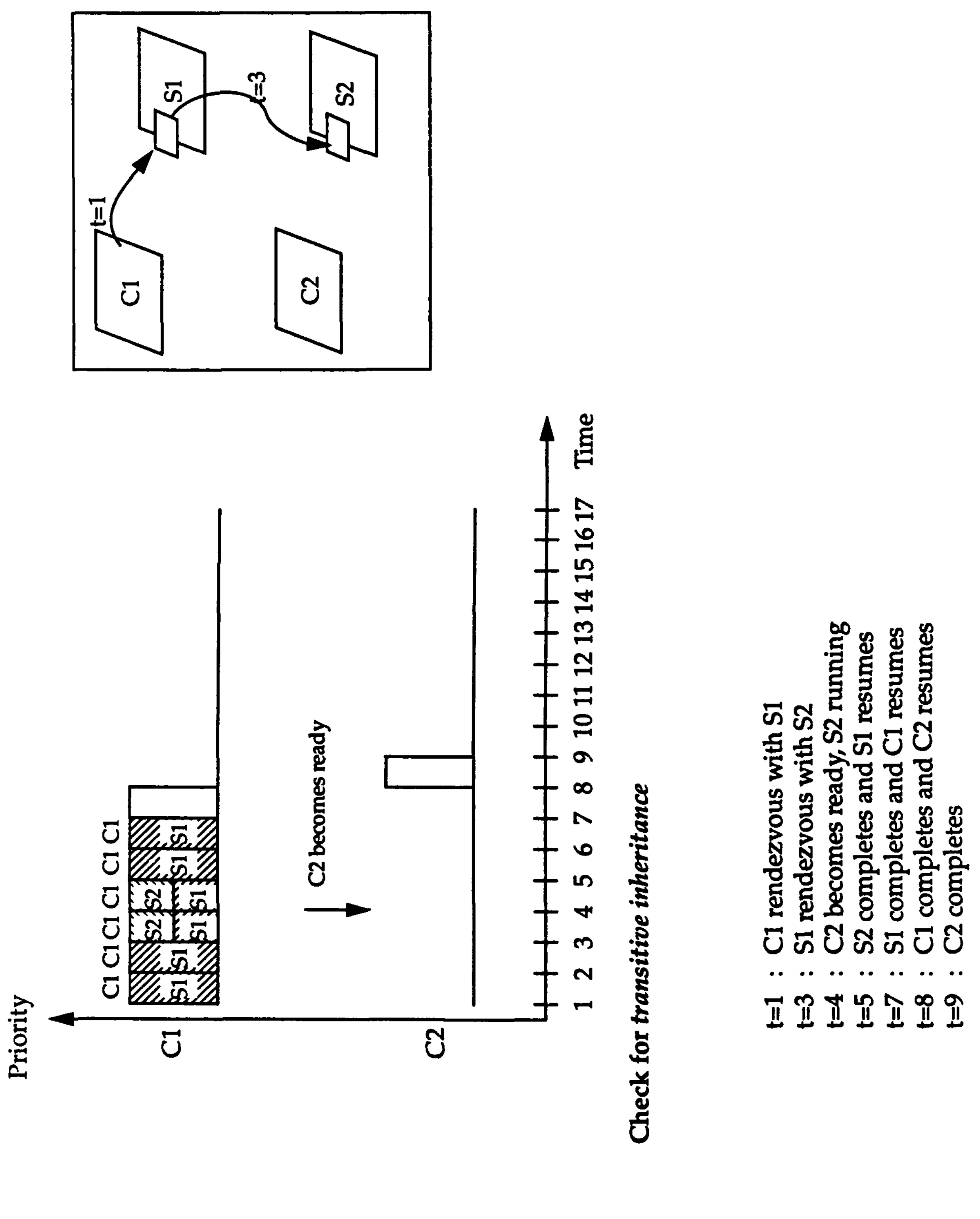
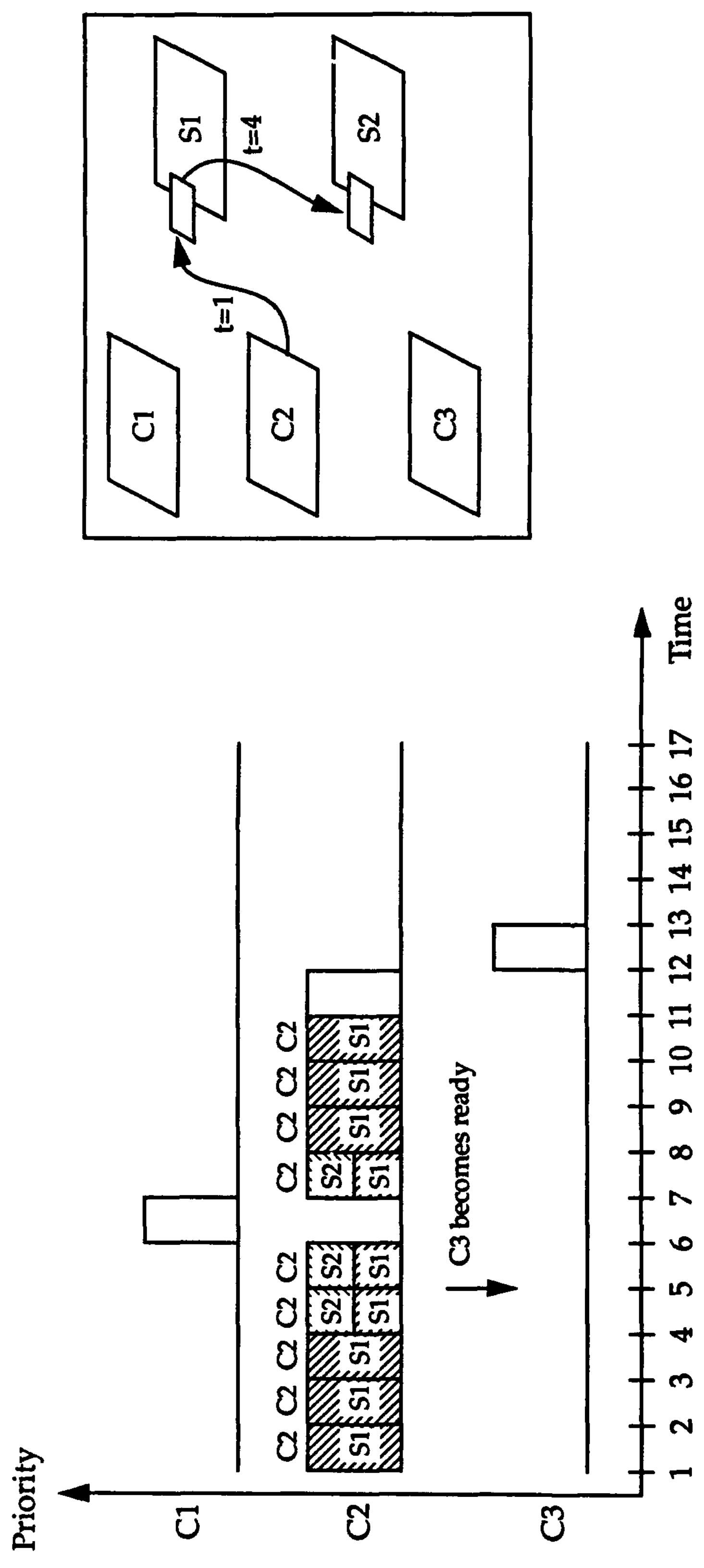

Priority
C1
C2
C1 C1 C1 C1 C1 C1 C1
S1 S1 S2 S2 S1 S1
S1 S1 S1 S1
C2 becomes ready
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Time
Check for *transitive inheritance*
t=1 : C1 rendezvous with S1
t=3 : S1 rendezvous with S2
t=4 : C2 becomes ready, S2 running
t=5 : S2 completes and S1 resumes
t=7 : S1 completes and C1 resumes
t=8 : C1 completes and C2 resumes
t=9 : C2 completes
C1
t=1
S1
t=3
S2
C2

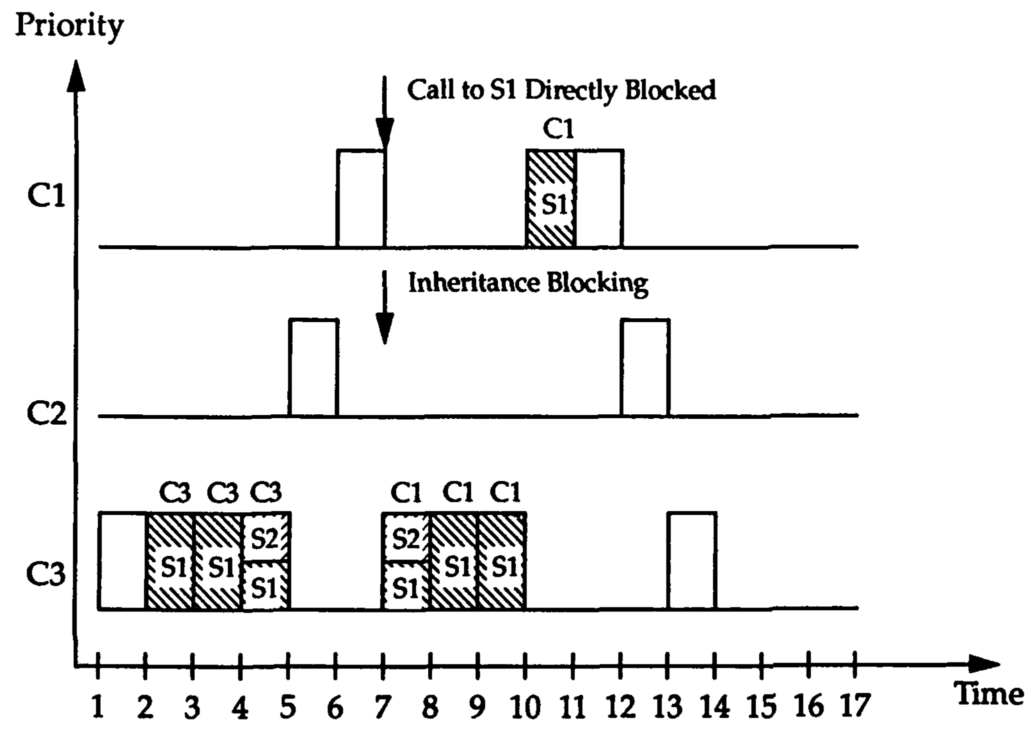
Figure A-7 BI_03: Inheritance Is Transitive

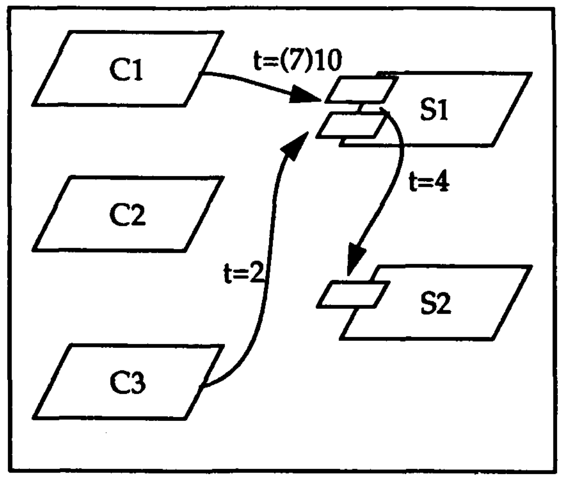


Check for *transitive inheritance* which allows higher priority preemption

t=1 : C2 rendezvous with S1
t=4 : S1 rendezvous with S2
t=5 : C3 becomes ready, S2 running
t=6 : C1 preempts S2
t=7 : C1 completes and S2 resumes
t=8 : S2 completes and S1 resumes
t=11 : S1 completes and C2 resumes
t=12 : C2 completes
C3 begins execution
t=13 : C3 completes

Figure A-8 BI_04: Transitive Inheritance with Preemption



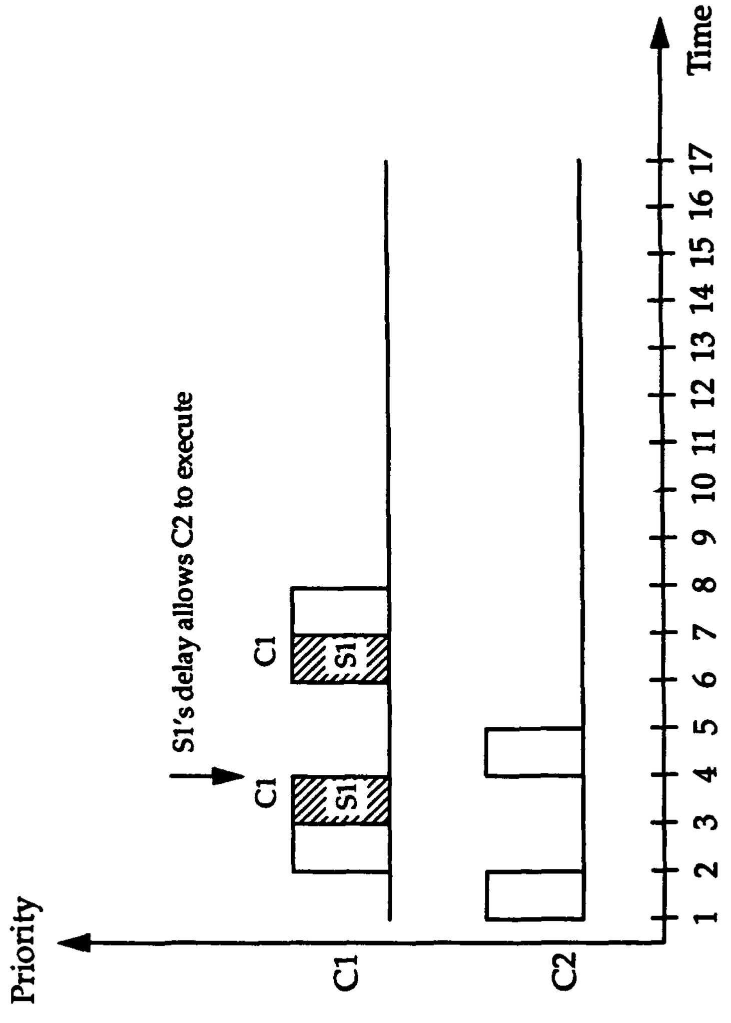


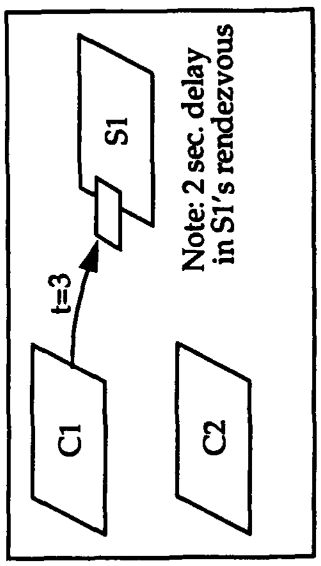
Check for *transitive inheritance* which causes blocking of preempted clients

t=1 : C3 begins execution
t=2 : C3 rendezvous with S1
t=4 : S1 rendezvous with S2
t=5 : C2 preempts S2
t=6 : C1 preempts C2
t=7 : C1 attempts to call S1 [DIRECTLY BLOCKING]
S2 resumes on behalf of C3

t=8 : S2 completes and S1 resumes
t=10: S1 completes
C1 resumes and rendezvous with S1
t=11: S1 completes and C1 resumes
t=12: C1 completes and C2 resumes
t=13: C2 completes and C3 resumes
t=14: C3 completes

Figure A-9 BI_05: Transitive Inheritance with Server at High Priority

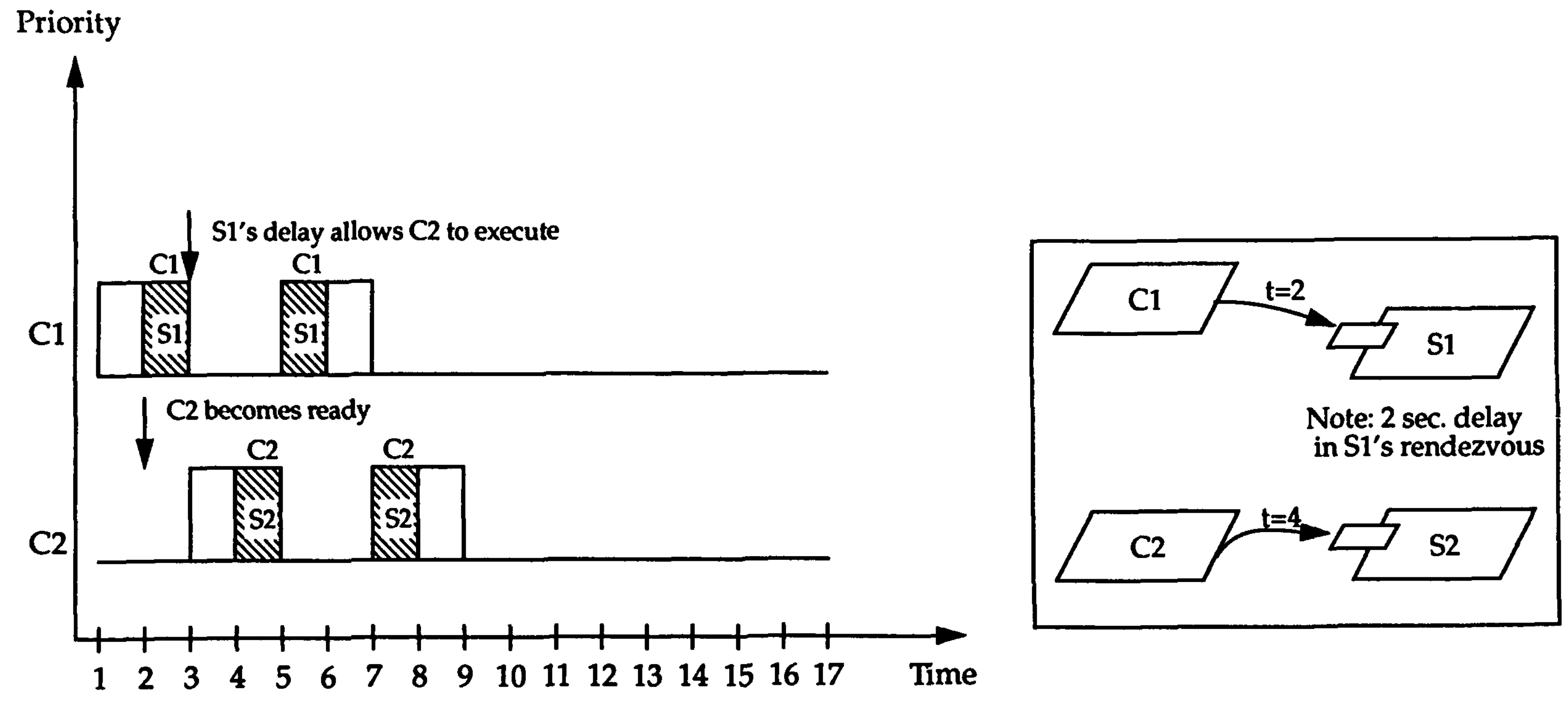




Check that *delayed servers don't block* lower-priority clients

t=1 : C2 begins execution
t=2 : C1 preempts C2
t=3 : C1 rendezvous with S1
t=4 : S1 is delayed
C2 resumes
t=5 : C2 completes
t=6 : S1 resumes
t=7 : S1 completes and C1 resumes
t=8 : C1 completes

Figure A-10 BI_06: Delayed Server Doesn't Block

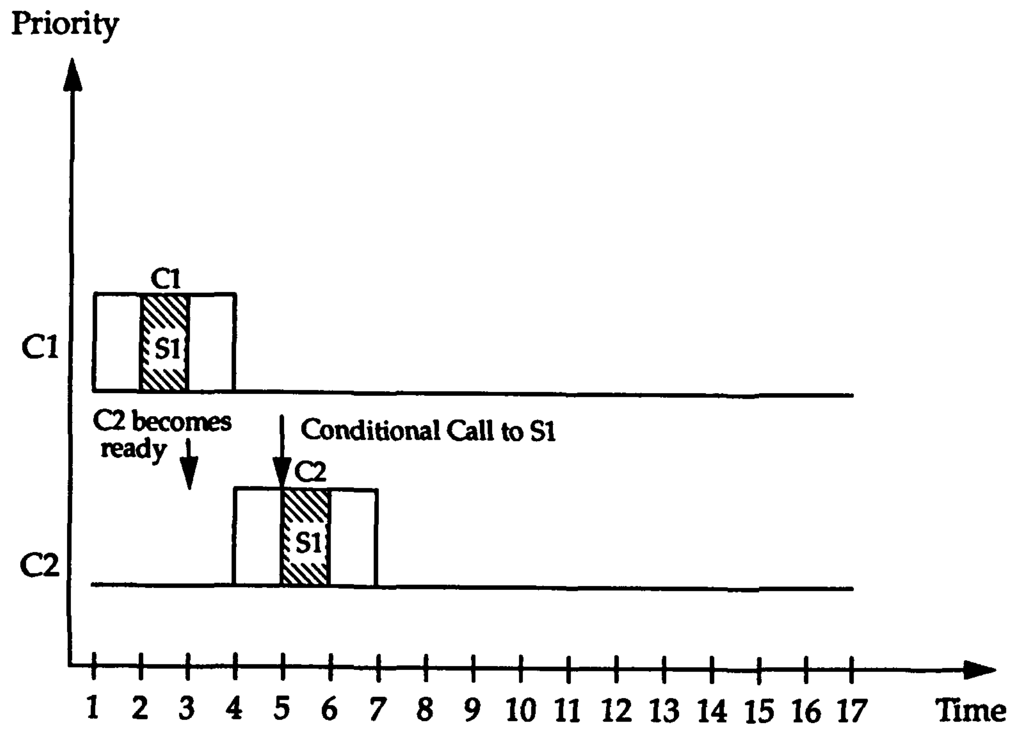


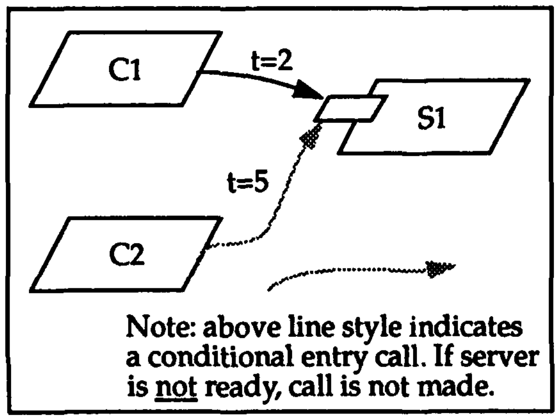
Check that *delayed servers don't block* lower-priority clients

t=1	:	C1 begins execution
t=2	:	C1 rendezvous with S1
		C2 becomes ready
t=3	:	S1 is delayed
		C2 begins execution
t=4	:	C2 rendezvous with S2
t=5	:	S1 resumes (delay complete)
t=6	:	S1 completes and C1 resumes
t=7	:	C1 completes and S2 resumes
t=8	:	S2 completes and C2 resumes
t=9	:	C2 completes

See PC_07 for behavior under PCP

Figure A-11 BI_07: Delayed Servers Don't Block




C1
t=2
S1
t=5
C2
Note: above line style indicates a conditional entry call. If server is not ready, call is not made.

Check that a server is ready to accept a call immediately after completing a rendezvous

t=1	:	C1 begins execution
t=2	:	C1 rendezvous with S1
t=3	:	S1 completes and C1 resumes C2 becomes ready
t=4	:	C1 completes C2 begins execution
t=5	:	C2 rendezvous with S1
t=6	:	S1 completes and C2 resumes
t=7	:	C2 completes

Figure A-12 BI_08: Immediate Rendezvous with Server

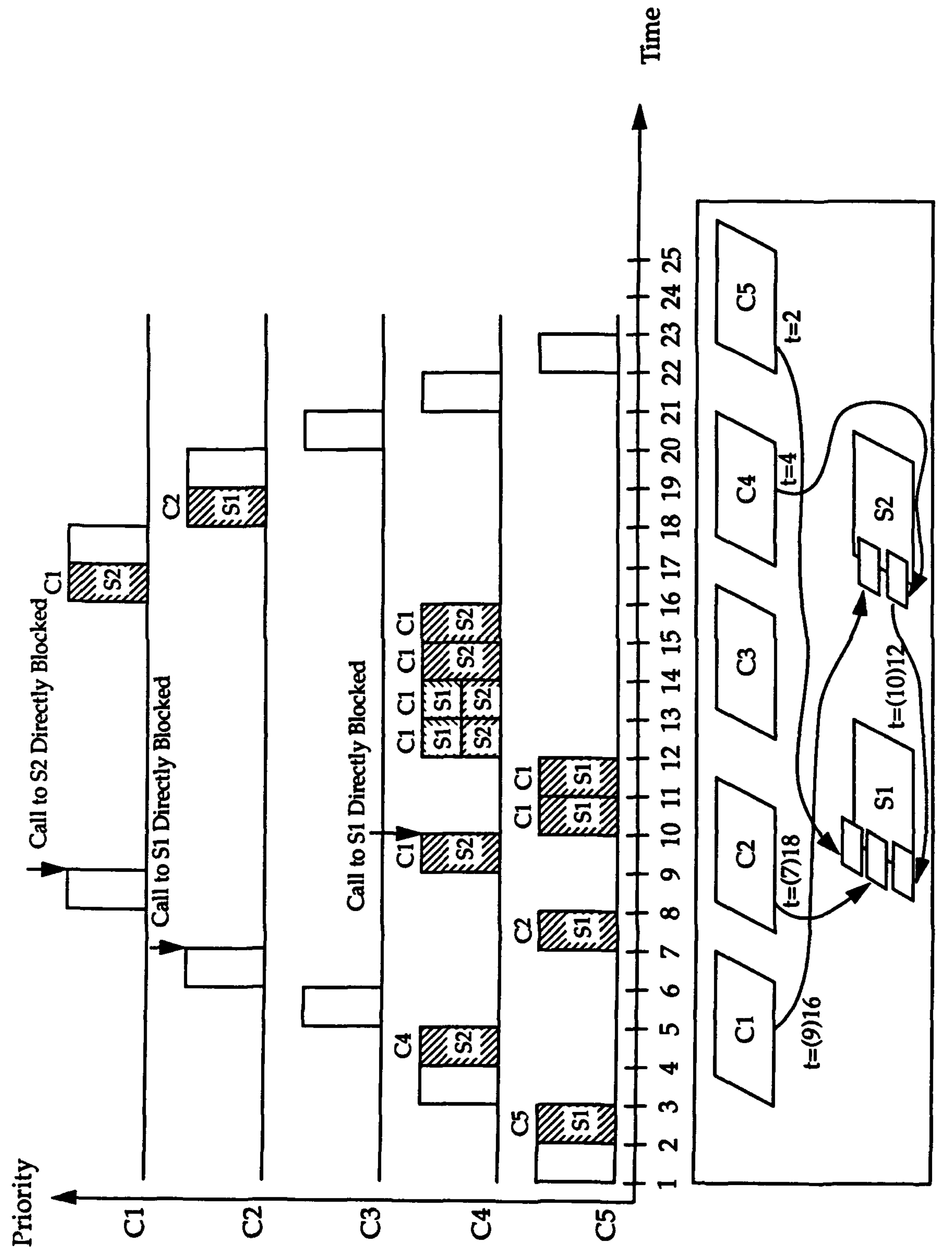


Figure A-13 BI_09: Complex Example #1

Check for preemption, blocking, inheritance, and proper priorities of tasks for a basic inheritance system. (Similar to [Borger 89] Example #1)

```
t=1  : C5 begins execution
t=2  : C5 rendezvous with S1
t=3  : C4 preempts C1
t=4  : C4 rendezvous with S2
t=5  : C3 preempts S2
t=6  : C2 preempts C3
t=7  : C2 attempts to call S1 [DIRECTLY BLOCKED]
       S1 resumes on behalf of C5
t=8  : C1 preempts S1
t=9  : C1 attempts to call S2 [DIRECTLY BLOCKED]
       S2 resumes on behalf of C4
t=10 : S2 attempts to call S1 [DIRECTLY BLOCKED]
       S1 resumes on behalf of C5
t=12 : S1 completes
       S2 resumes and rendezvous with S1
t=14 : S1 completes and S2 resumes
t=16 : S2 completes
       C1 resumes and rendezvous with S2
t=17 : S2 completes and C1 resumes
t=18 : C1 completes
       C2 resumes and rendezvous with S1
t=19 : S1 completes and C2 resumes
t=20 : C2 completes and C3 resumes
t=21 : C3 completes and C4 resumes
t=22 : C4 completes and C5 resumes
t=23 : C5 completes
```

Figure A-14 BI_09 Events

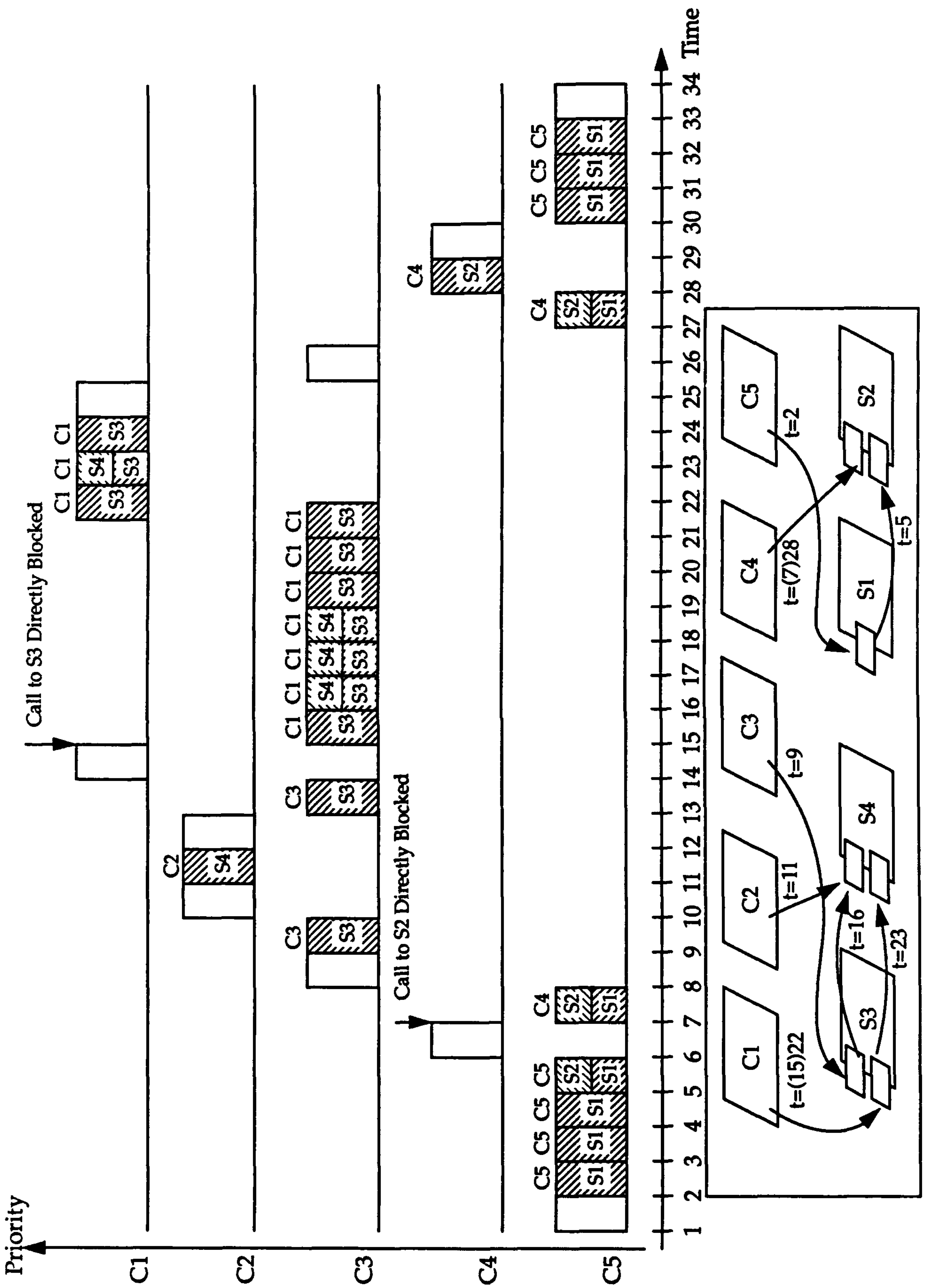
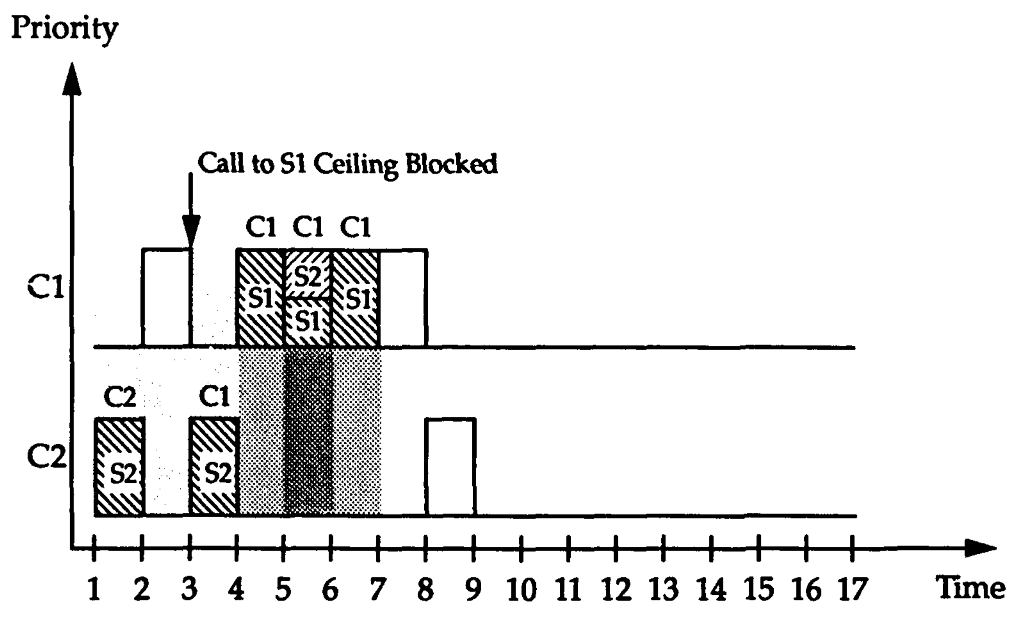


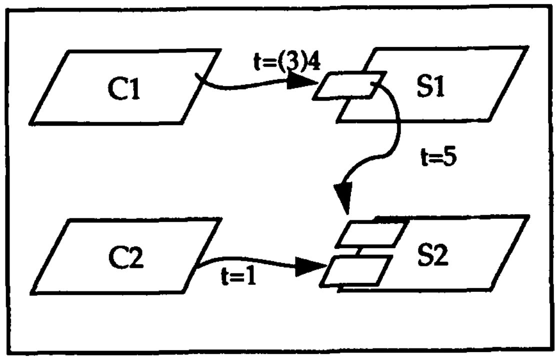
Figure A-15 BI_10: Complex Example #2

Check for preemption, blocking, inheritance, and proper priorities of tasks for a basic inheritance system. (Similar to [Borger 89] Example #2)

t=1 :	C5 begins execution
t=2 :	C5 rendezvous with S1
t=5 :	S1 rendezvous with S2
t=6 :	C4 preempts S2
t=7 :	C4 attempts to call S2 [DIRECTLY BLOCKED]
	S2 resumes on behalf of C5
t=8 :	C3 preempts S2
t=9 :	C3 rendezvous with S3
t=10 :	C2 preempts S3
t=11 :	C2 rendezvous with S4
t=12 :	S4 completes and C2 resumes
t=13 :	C2 completes and S3 resumes
t=14 :	C1 preempts S3
t=15 :	C1 attempts to call S3 [DIRECTLY BLOCKED]
	S3 resumes on behalf of C3
t=16 :	S3 rendezvous with S4
t=19 :	S4 completes and S3 resumes
t=22 :	S3 completes
	C1 resumes and rendezvous with S3
t=23 :	S3 rendezvous with S4
t=24 :	S4 completes and S3 resumes
t=25 :	S3 completes and C1 resumes
t=26 :	C1 completes and C3 resumes
t=27 :	C3 completes and S2 resumes
t=28 :	S2 completes
	C2 resumes and rendezvous with S2
t=29 :	S2 completes and C4 resumes
t=30 :	C4 rendezvous with S1
t=33 :	S1 completes and C5 resumes
t=34 :	C5 completes

Figure A-16 BI_10 Events



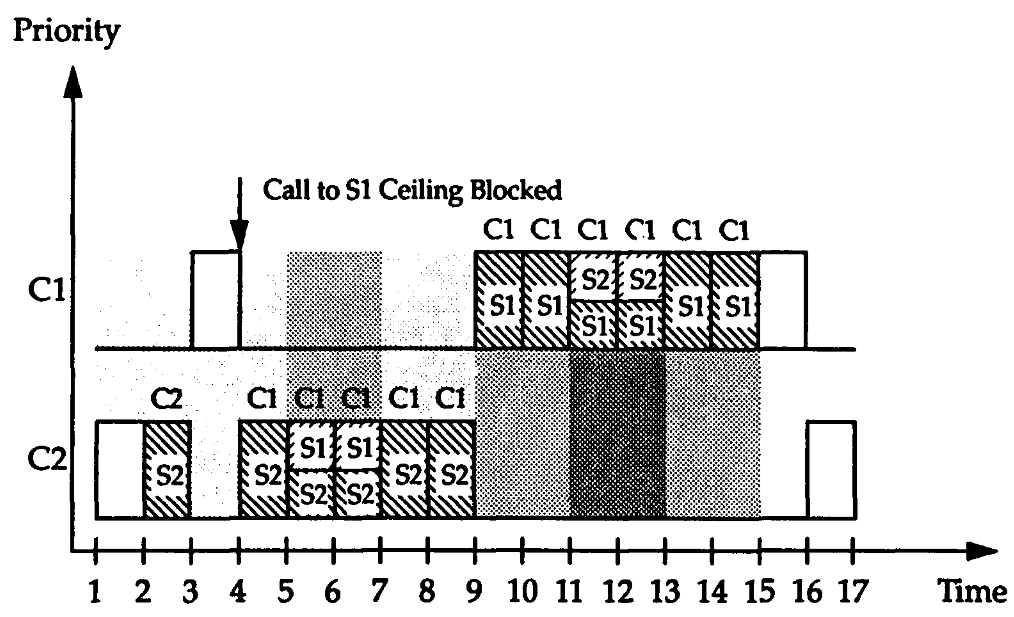


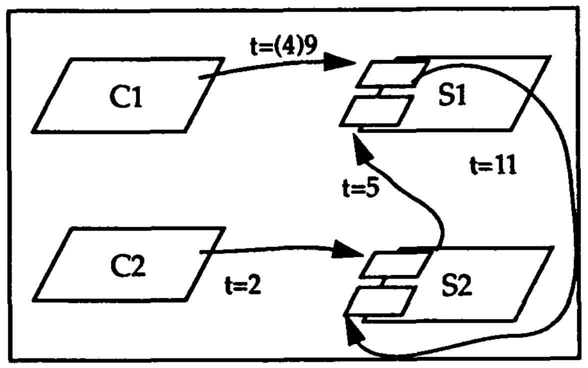
Check for *Ceiling Blocking*

t=1 : C2 rendezvous with S2
t=2 : C1 preempts S2
t=3 : C1 attempts to call S1 [CEILING BLOCKED]
S2 resumes on behalf of C2
t=4 : S2 completes
C1 preempts C2 and rendezvous with S1
t=5 : S1 rendezvous with S2
t=6 : S2 completes and S1 resumes
t=7 : S1 completes and C1 resumes
t=8 : C1 completes and C2 resumes
t=9 : C2 completes

Under either emulation method, C1 will not execute until S2 has completed its rendezvous with C2.

Figure A-17 PC_01: Simple Ceiling Blocking



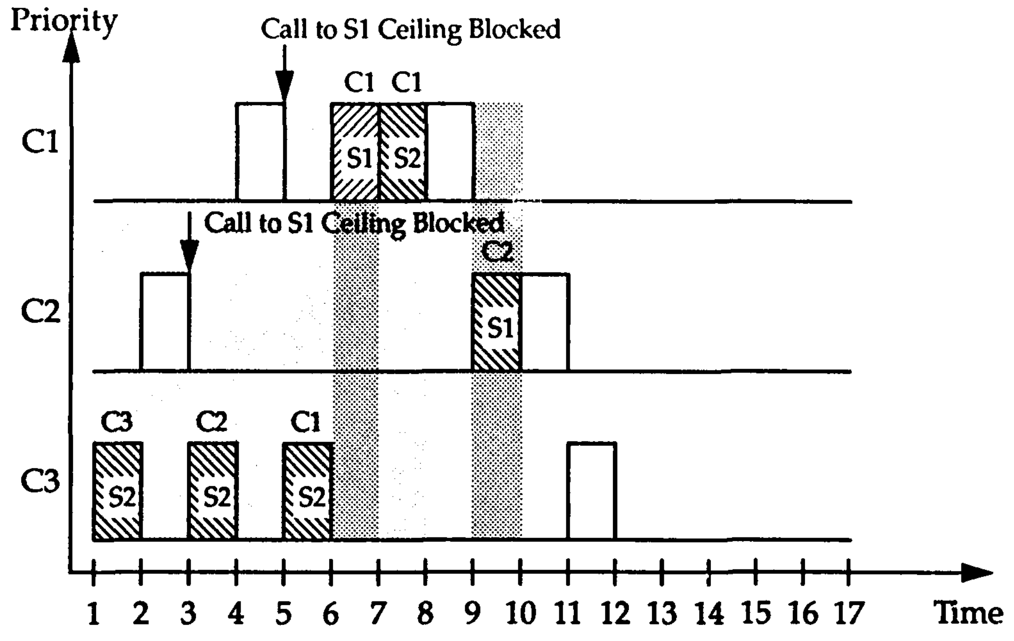


Under either emulation method, C1 will not execute until S2 has completed its rendezvous with C2.

Check for *Deadlock avoidance* (2 tasks)

t=1 :	C2 begins to execute	t=9 :	S2 completes
t=2 :	C2 rendezvous with S2		C1 preempts C2 and rendezvous with S1
t=3 :	C1 preempts S2	t=11 :	S1 rendezvous with S2
t=4 :	C1 attempts to call S1 [CEILING BLOCKED]	t=13 :	S2 completes and S1 resumes
	S2 resumes on behalf of C2	t=15 :	S1 completes and C1 resumes
t=5 :	S2 rendezvous with S1	t=16 :	C1 completes and C2 resumes
t=7 :	S1 completes and S2 resumes	t=17 :	C2 completes

Figure A-18 PC_02: Deadlock Avoidance (2 tasks)



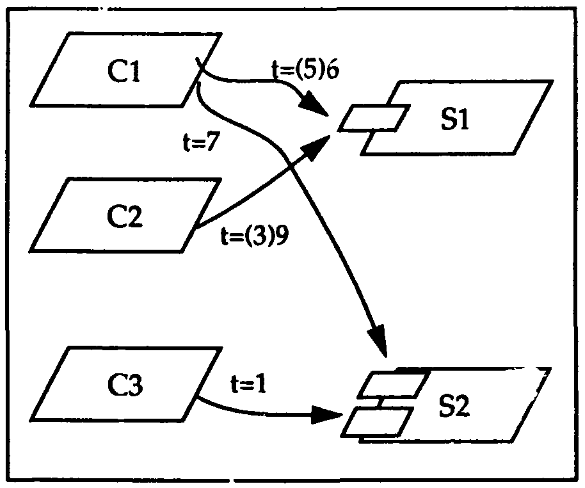


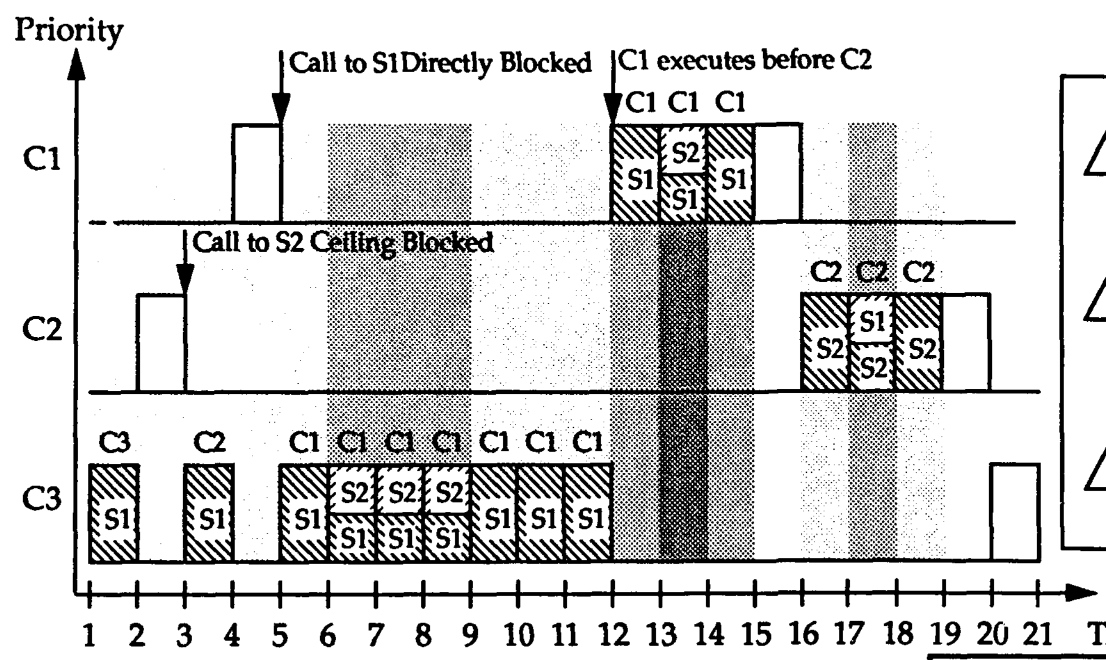
Figure A-19 PC_03: Blocked at Most Once

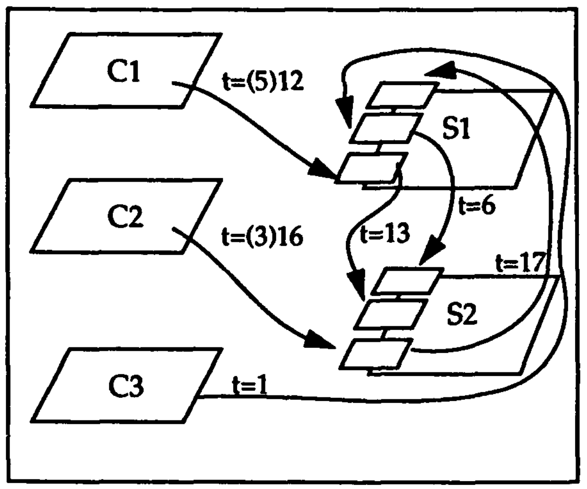
Check that higher-priority clients are *blocked at most ONCE*

t=1 : C3 rendezvous with S2
t=2 : C2 preempts S2
t=3 : C2 attempts to call S1 [CEILING BLOCKED]
S2 resumes on behalf of C3
t=4 : C1 preempts S2
t=5 : C1 attempts to call S1 [CEILING BLOCKED]
S2 resumes on behalf of C3
t=6 : S2 completes
C1 preempts C3 & C2 and rendezvous with S1

t=7 : S1 completes
C1 rendezvous with S2
t=8 : S2 completes and C1 resumes
t=9 : C1 completes
C2 preempts C3 and rendezvous with S1
t=10: S1 completes and C2 resumes
t=11: C2 completes and C3 resumes
t=12: C3 completes

Under either emulation method, C1 and C2 will not execute until S2 has completed its rendezvous with C3.



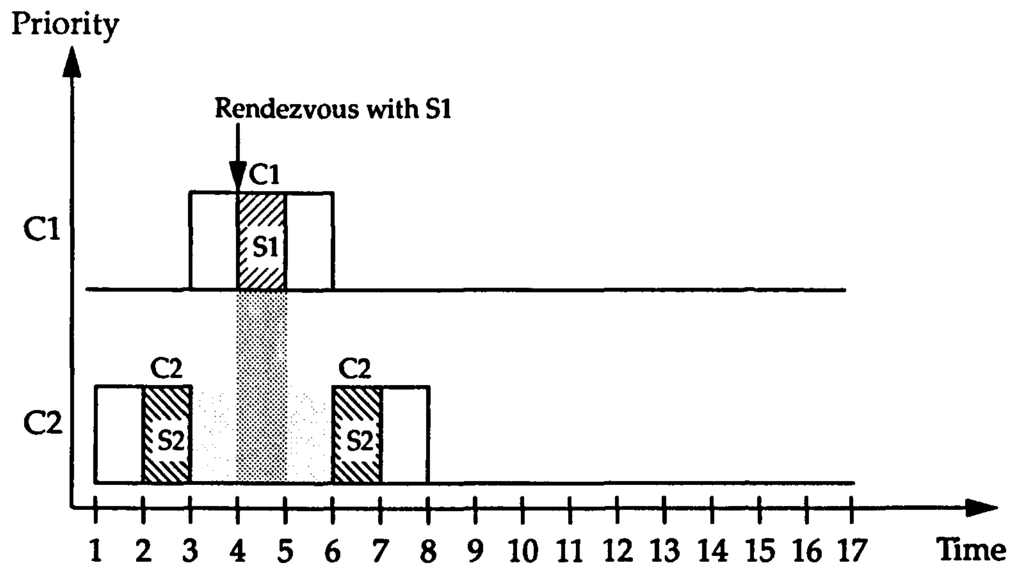


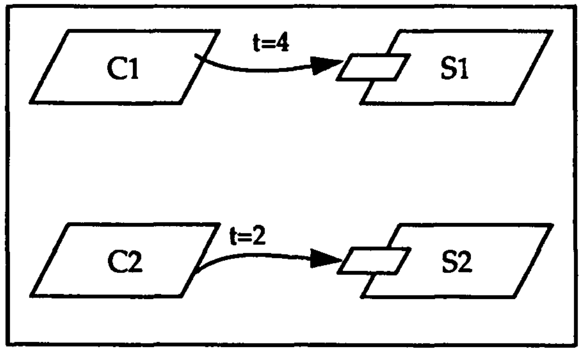
Under either emulation method, neither C1 nor C2 execute until S1 completes its rendezvous with C3.

Check for *blocked at most once* by lower client's rendezvous

t=1 :	C3 rendezvous with S1
t=2 :	C2 preempts S1
t=3 :	C2 attempts to call S2 [CEILING BLOCKED]
	S1 resumes on behalf of C3
t=4 :	C1 preempts S1
t=5 :	C1 attempts to call S1 [DIRECTLY BLOCKED]
	S1 resumes on behalf of C3
t=6 :	S1 rendezvous with S2
t=9 :	S2 completes and S1 resumes
t=12:	S1 completes
	C1 preempts C3 & C2 and rendezvous with S1
t=13:	S1 rendezvous with S2
t=14:	S2 completes and S1 resumes
t=15:	S1 completes and C1 resumes
t=16:	C1 completes
	C2 preempts C3 and rendezvous with S2
t=17:	S2 rendezvous with S1
t=18:	S1 completes and S2 resumes
t=19:	S2 completes and C2 resumes
t=20:	C2 completes and C3 resumes
t=21:	C3 completes

Figure A-20 PC_04: Blocked at Most Once (Nested Calls)



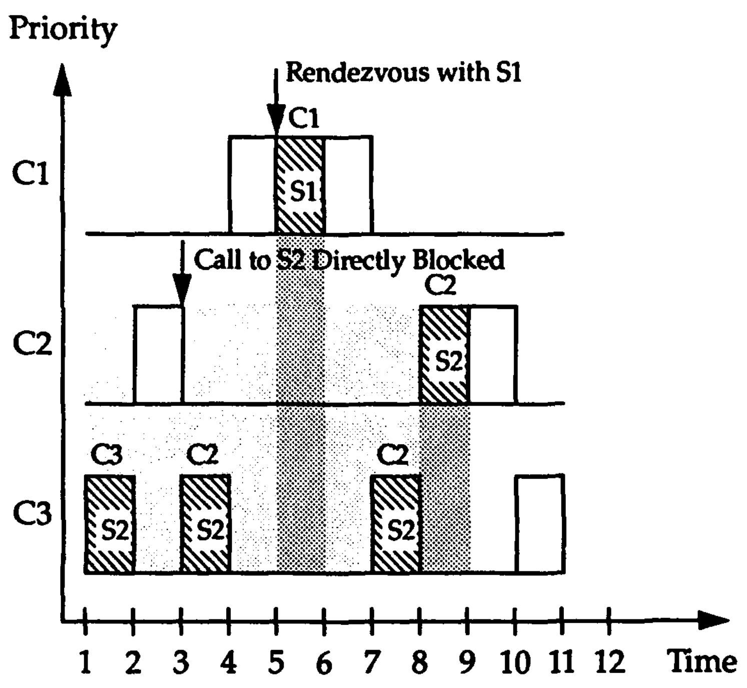


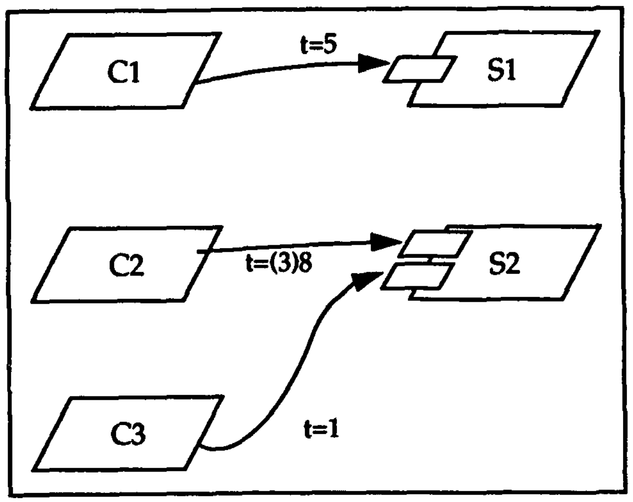
Check that clients with priority above preempted server's ceiling *can rendezvous* with ready servers

t=1	:	C2 begins execution
t=2	:	C2 rendezvous with S2
t=3	:	C1 preempts S2
t=4	:	C1 rendezvous with S1
t=5	:	S1 completes and C1 resumes
t=6	:	C1 completes and S2 resumes
t=7	:	S2 completes and C2 resumes
t=8	:	C2 completes

Under the non-preemptible-server emulation method, C1 will not execute until C2 completes its rendezvous.

Figure A-21 PC_05: Servers Don't Execute Above Ceiling Priority



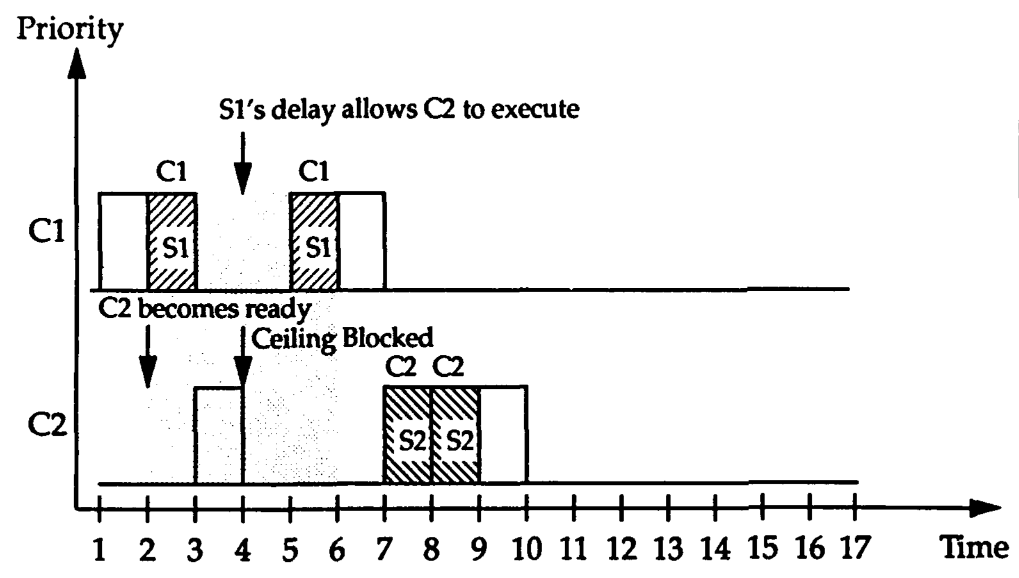


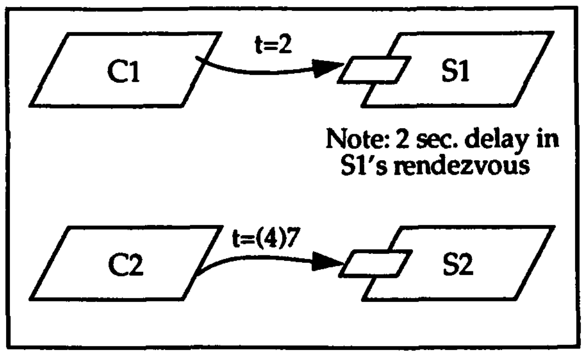
Check that clients with priority above preempted server's ceiling *can rendezvous* with ready servers and correct task resumes

t=1	:	C3 rendezvous with S2
t=2	:	C2 preempts S2
t=3	:	C2 attempts to call S2 [DIRECTLY BLOCKED]
		S2 resumes on behalf of C3
t=4	:	C1 preempts S2
t=5	:	C1 rendezvous with S1
t=6	:	S1 completes and C1 resumes
t=7	:	C1 completes and S2 resumes
t=8	:	S2 completes
		C2 preempts C3 and rendezvous with S2
t=9	:	S2 completes and C2 resumes
t=10	:	C2 completes and C3 resumes
t=11	:	C3 completes

Under the server-ceiling emulation method, C2 will not execute until C3's rendezvous is complete. Under the non-preemptible-server emulation method, both C1 and C2 will not execute until C3's rendezvous is complete.

Figure A-22 PC_06: Correct Task Resumes



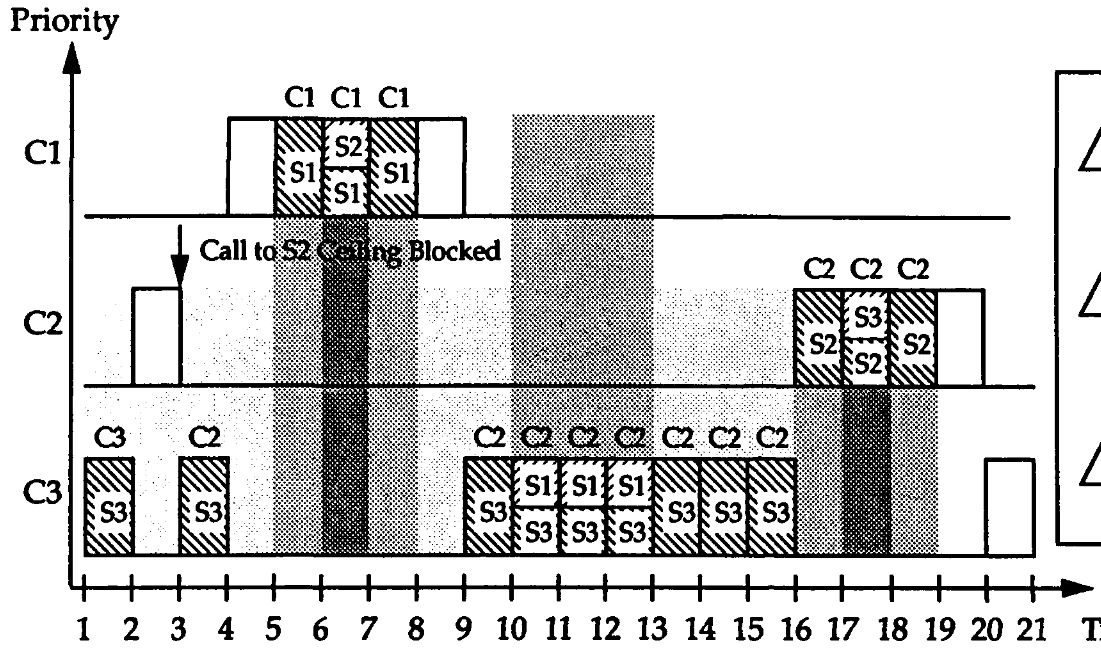


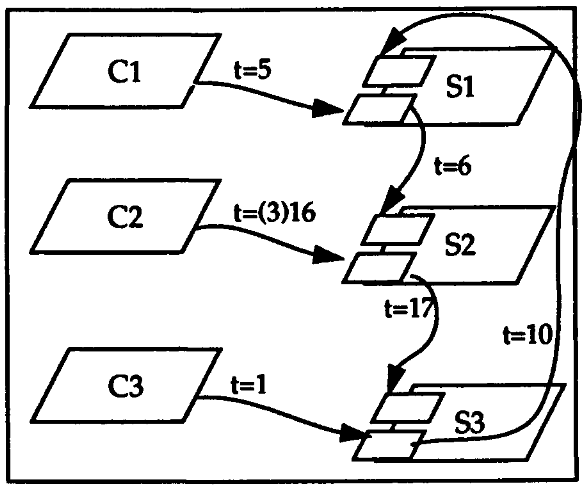
Check that *delayed servers don't block* lower-priority clients

t=1	:	C1 begins execution
t=2	:	C1 rendezvous with S1
		C2 becomes ready
t=3	:	S1 is delayed
		C2 begins execution
t=4	:	C2 attempts to call S2 [CEILING BLOCKED]
t=5	:	S1 resumes (delay complete)
t=6	:	S1 completes and C1 resumes
t=7	:	C1 completes and C2 resumes
		C2 rendezvous with S2
t=9	:	S2 completes and C2 resumes
t=10	:	C2 completes

Under the non-preemptible-server emulation method, C1 will not execute until S1's rendezvous is complete. Under the server-ceiling emulation method, C2 will not be blocked from calling S2. This could lead to deadlock or chained blocking (e.g., if C1 accessed S2, and if C2 accessed S1).

Figure A-23 PC_07: Delayed High-Priority Server Doesn't Block



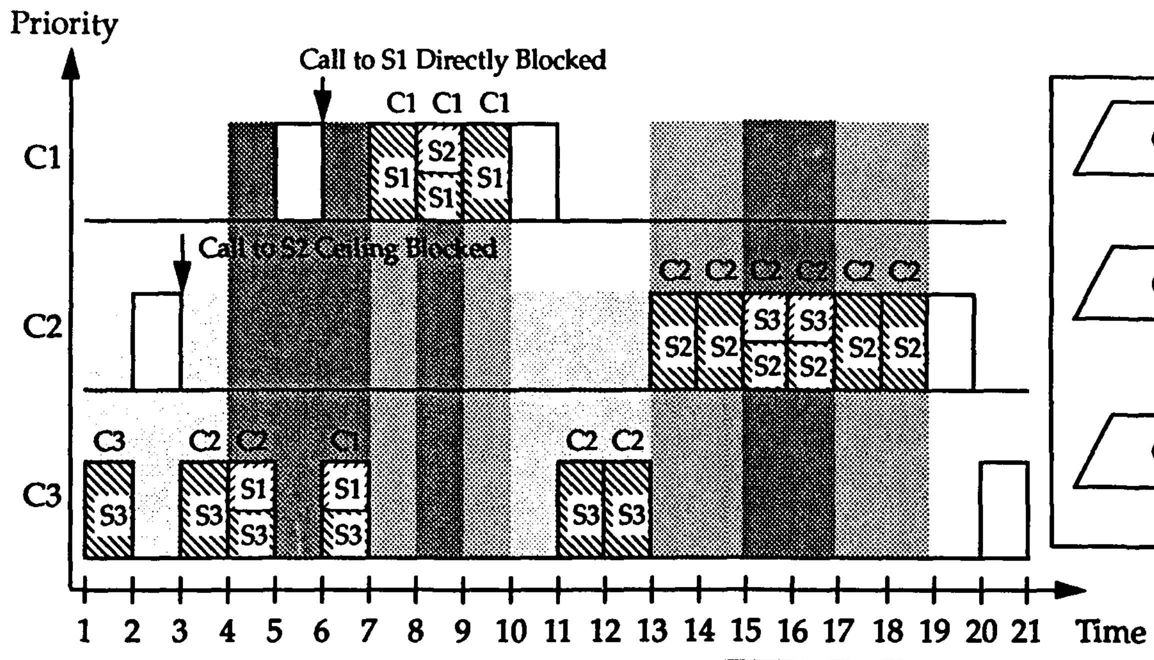


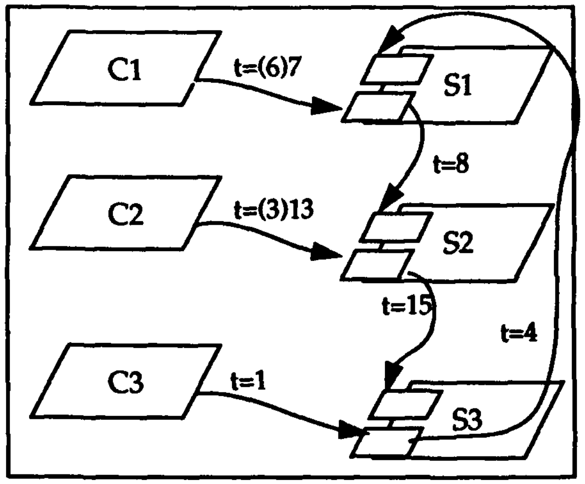
Under the server-ceiling emulation method, C2 will not execute until S3's rendezvous with C3 is complete. Under the non-preemptible-server emulation method, neither C1 nor C2 will execute

Check for *deadlock avoidance* (3 tasks)

t=1 :	C3 rendezvous with S3
t=2 :	C2 preempts S3
t=3 :	C2 attempts to call S2 [CEILING BLOCKED] S3 resumes on behalf of C3
t=4 :	C1 preempts S3
t=5 :	C1 rendezvous with S1
t=6 :	S1 rendezvous with S2
t=7 :	S2 completes and S1 resumes
t=8 :	S1 completes and C1 resumes
t=9 :	C1 completes and S3 resumes
t=10:	S3 rendezvous with S1
t=13:	S1 completes and S3 resumes
t=16:	S3 completes C2 preempts C3 and rendezvous with S2
t=17:	S2 rendezvous with S3
t=18:	S3 completes and S2 resumes
t=19:	S2 completes and C2 resumes
t=20:	C2 completes and C3 resumes
t=21:	C3 completes

Figure A-24 PC_08: Deadlock Avoidance (3 Tasks)





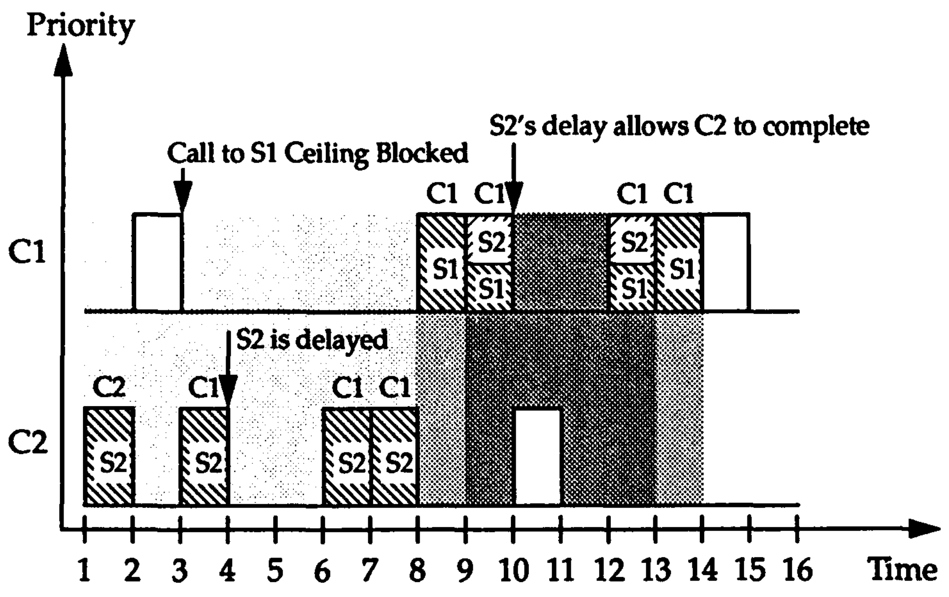
Under either emulation method, neither C1 nor C2 will be allowed to execute until C3 has completed its rendezvous with S3. Then C1 will execute first.

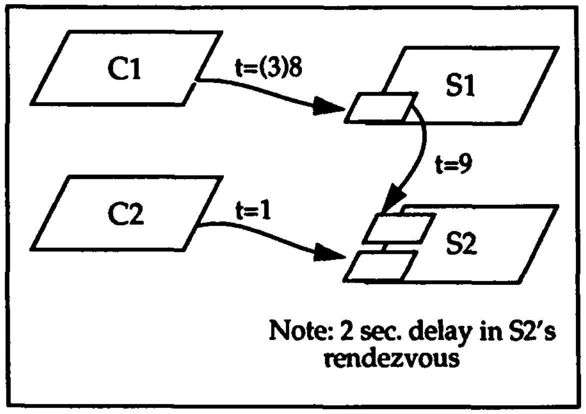
Check for *deadlock avoidance* (3 tasks)

t=1 : C3 rendezvous with S3
t=2 : C2 preempts S3
t=3 : C2 attempts to call S2 [CEILING BLOCKED]
S3 resumes on behalf of C3
t=4 : S3 rendezvous with S1
t=5 : C1 preempts S1
t=6 : C1 attempts to call S1 [DIRECTLY BLOCKED]
S1 resumes on behalf of C3
t=7 : S1 completes
C1 preempts and rendezvous with S1

t=8 : S1 rendezvous with S2
t=9 : S2 completes and S1 resumes
t=10: S1 completes and C1 resumes
t=11: C1 completes and S3 resumes
t=13: S3 completes
C2 preempts C3 and rendezvous with S2
t=15: S2 rendezvous with S3
t=17: S3 completes and S2 resumes
t=19: S2 completes and C2 resumes
t=20: C2 completes and C3 resumes
t=21: C3 completes

Figure A-25 PC_09: Deadlock Avoidance (3 Tasks)





Check that when a server is delayed, a ceiling-blocked call is still blocked and delays don't block ready clients

t=1 : C2 rendezvous with S2
t=2 : C1 preempts S2
t=3 : C1 attempts to call S1 [CEILING BLOCKED]
S2 resumes on behalf of C2
t=4 : S2 delays
No clients ready to execute
t=6 : S2 resumes
t=8 : S2 completes
C1 preempts C2 and rendezvous with S1
t=9 : S1 rendezvous with S2
t=10 : S2 delays
C2 resumes
t=11 : C2 completes
No clients ready to execute
t=12 : S2 resumes
t=13 : S2 completes and S1 resumes
t=14 : S1 completes and C1 resumes
t=15 : C1 completes

Under the server-ceiling emulation method, C1 will execute when S2 is delayed until C1's call to S2 is blocked. Under the non-preemptible-server emulation method, C1 will not be allowed to execute until C2's rendezvous is complete.

Figure A-26 PC_10: Ceiling Blocking Works When Server Is Delayed

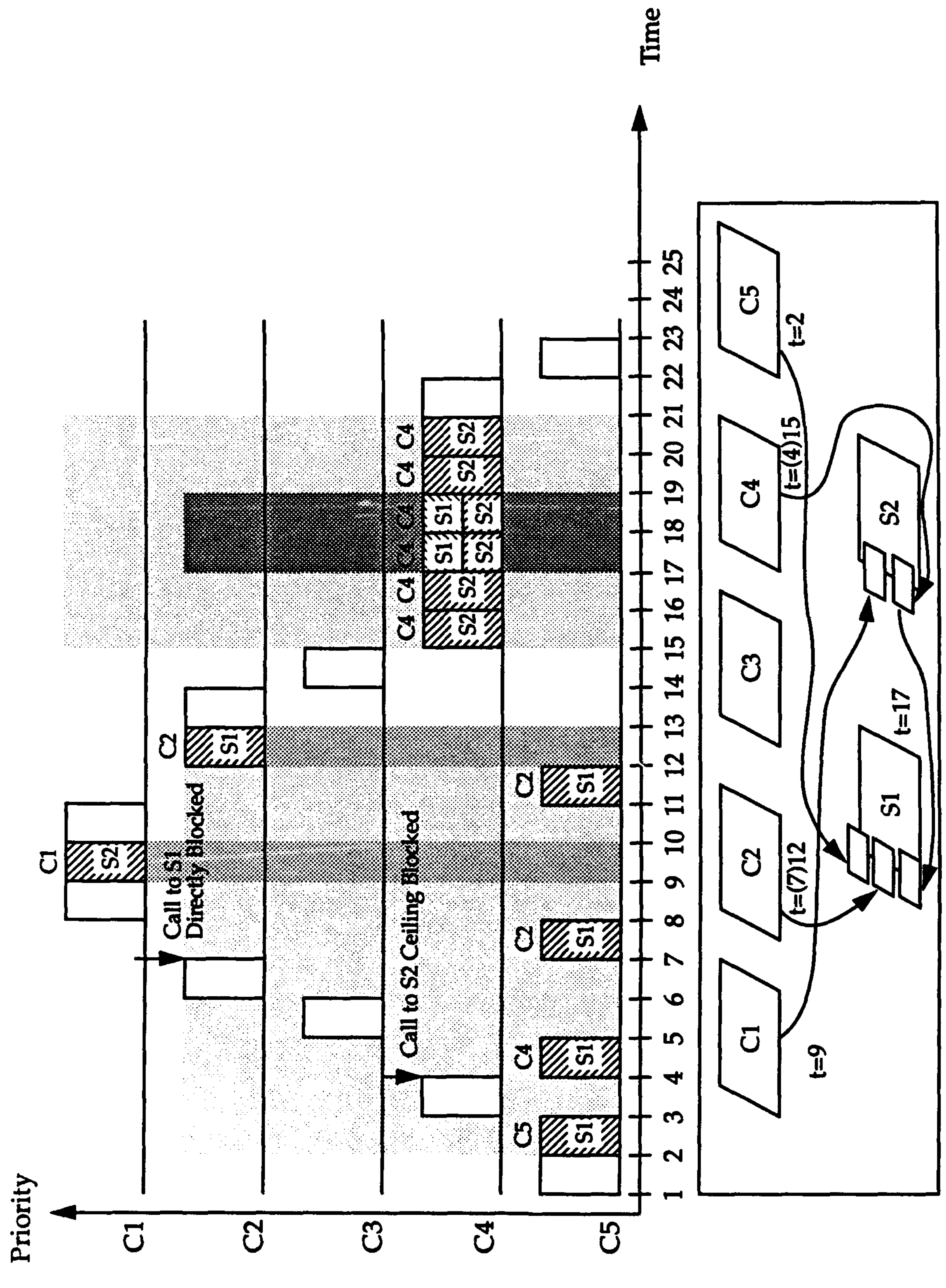


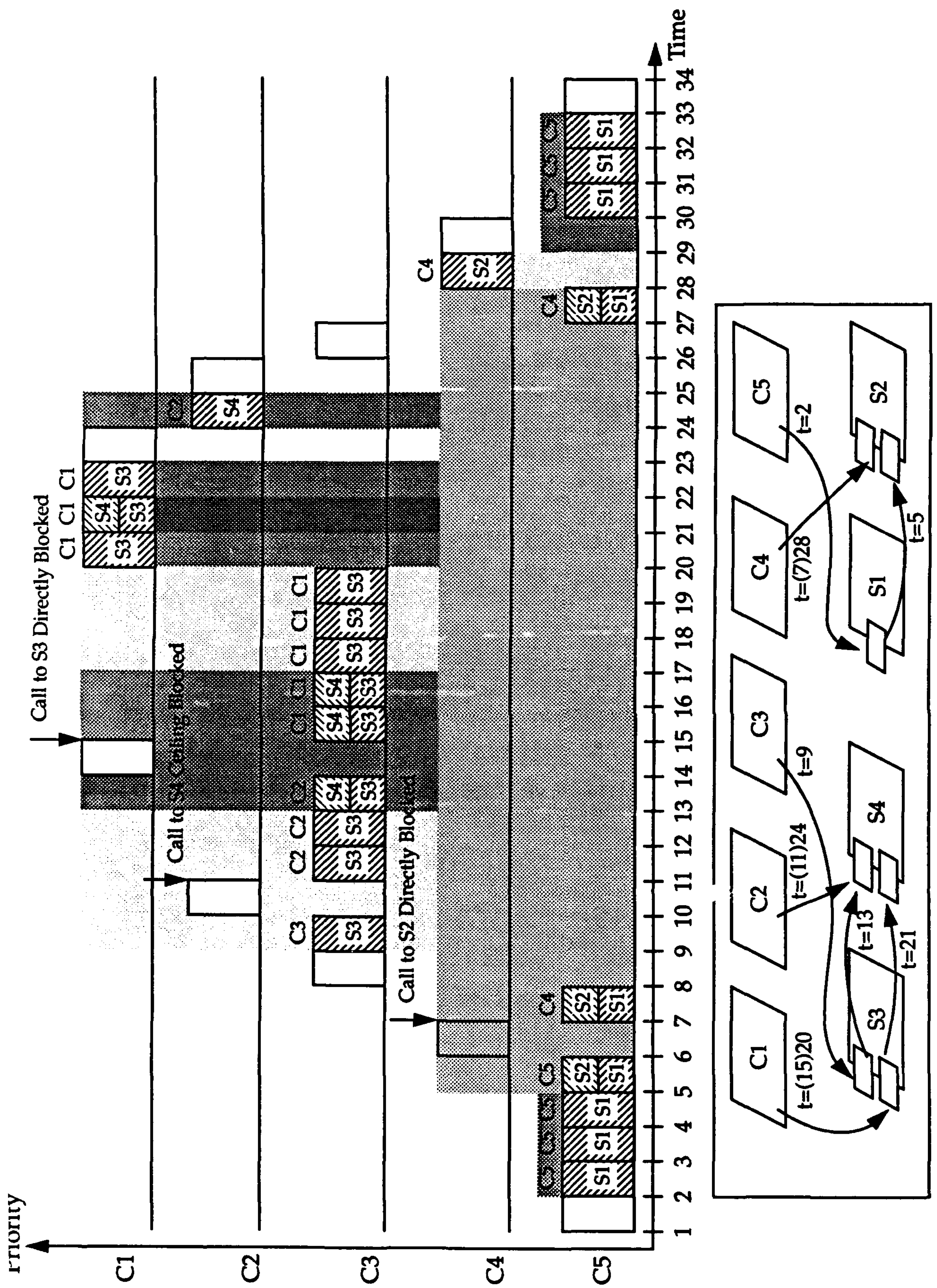
Figure A-27 PC_11: Complex Example #1

Check for preemption, blocking, inheritance, and proper priorities of tasks for a Priority Ceiling Protocol system.
(Similar to [Borger 89] Example #1)

t=1 :	C5 begins execution
t=2 :	C5 rendezvous with S1
t=3 :	C4 preempts C1
t=4 :	C4 attempts to call S2[CEILING BLOCKED]
	S1 resumes on behalf of C5
t=5 :	C3 preempts S1
t=6 :	C2 preempts C3
t=7 :	C2 attempts to call S1 [DIRECTLY BLOCKED]
	S1 resumes on behalf of C5
t=8 :	C1 preempts S1
t=9 :	C1 rendezvous with S2
t=10 :	S2 completes and C1 resumes
t=11 :	C1 completes and S1 resumes
t=12 :	S1 completes
	C2 resumes and rendezvous with S1
t=13 :	S1 completes and C2 resumes
t=14 :	C2 completes and C3 resumes
t=15 :	C3 completes
	C4 resumes and rendezvous with S2
t=17 :	S2 rendezvous with S1
t=19 :	S1 completest and S2 resumes
t=21 :	S2 completes and C4 resumes
t=22 :	C4 completes and C4 resumes
t=23 :	C5 completes

Under the server-ceiling emulation method, C2, C3, and C4 are not permitted to start executing while C5's rendezvous with S1 is in progress. Under the non-preemptible-server emulation method, C1, C2, C3, and C4 are not permitted to execute until C5's rendezvous with S1 is complete.

Figure A-28 PC_11: Events



Priority
Time
C1
C2
C3
C4
C5
Call to S3 Directly Blocked
Call to S4 Ceiling Blocked
Call to S2 Directly Blocked
S1
S2
S3
S4
t=(15)20
t=(11)24
t=9
t=(7)28
t=2
t=13
t=21
t=5

Figure A-29 PC_12: Complex Example #2

Check for preemption, blocking, inheritance, and proper priorities of tasks for a Priority Ceiling Protocol system.

(Similar to [Borger 89] Example #2)

```
t=1  :  C5 begins execution                      t=20:  S3 completes
t=2  :  C5 rendezvous with S1                            C1 resumes and rendezvous with S3
t=5  :  S1 rendezvous with S2                    t=21:  S3 rendezvous with S4
t=6  :  C4 preempts S2                           t=22:  S4 completes and S3 resumes
t=7  :  C4 attempts to call S2 [DIRECTLY BLOCKED] t=23:  S3 completes and C1 resumes
        S2 resumes on behalf of C5               t=24:  C1 completes
t=8  :  C3 preempts S2                                   C2 resumes and rendezvous with S4
t=9  :  C3 rendezvous with S3                    t=25:  S4 completes and C2 resumes
t=10:  C2 preempts S3                            t=26:  C2 completes and C3 resumes
t=11 :  C2 attempts to call S4 [CEILING BLOCKED] t=27:  C3 completes and S2 resumes
        S3 resumes on behalf of C3               t=28:  S2 completes
t=13:  S3 rendezvous with S4                             C4 resumes and rendezvous with S2
t=14:  C1 preempts S4                            t=29:  S2 completes and C4 resumes
t=15:  C1 attempts to call S3 [DIRECTLY BLOCKED] t=30:  C4 completes and S1 resumes
        S4 resumes on behalf of S3               t=33:  S1 completes and C5 resumes
t=17:  S4 completes and S3 resumes               t=34:  C5 completes
```

Under the non-preemptible-server emulation method, higher-priority clients are not permitted to execute until C5 has completed its rendezvous with S1. The time line for this case is shown on the next page. The time line for the server-ceiling emulation method is shown on page 59.

Figure A-30 PC_12: Events

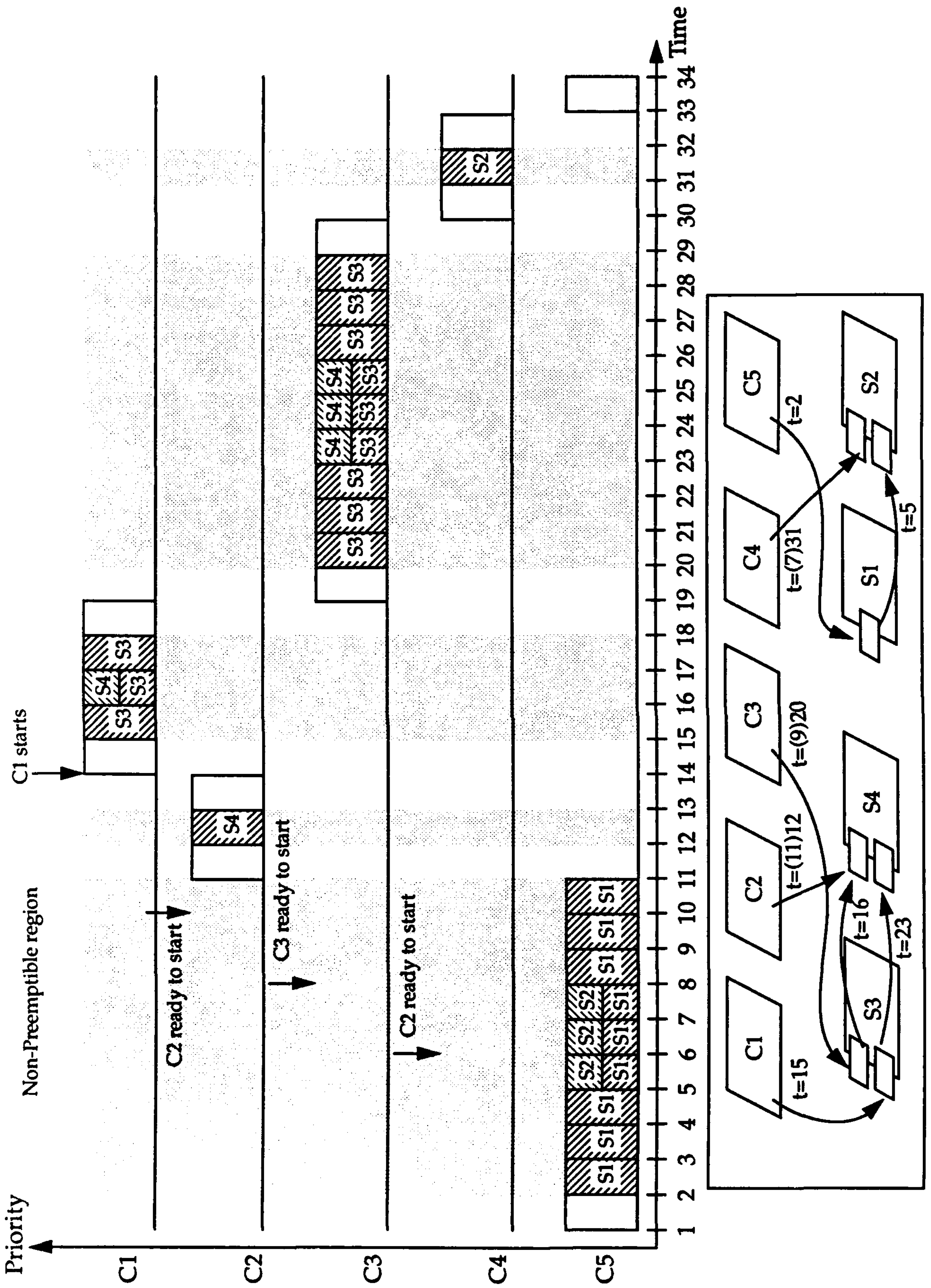


Figure A-31 PC_12: Events When Servers Are Non-Preemptible

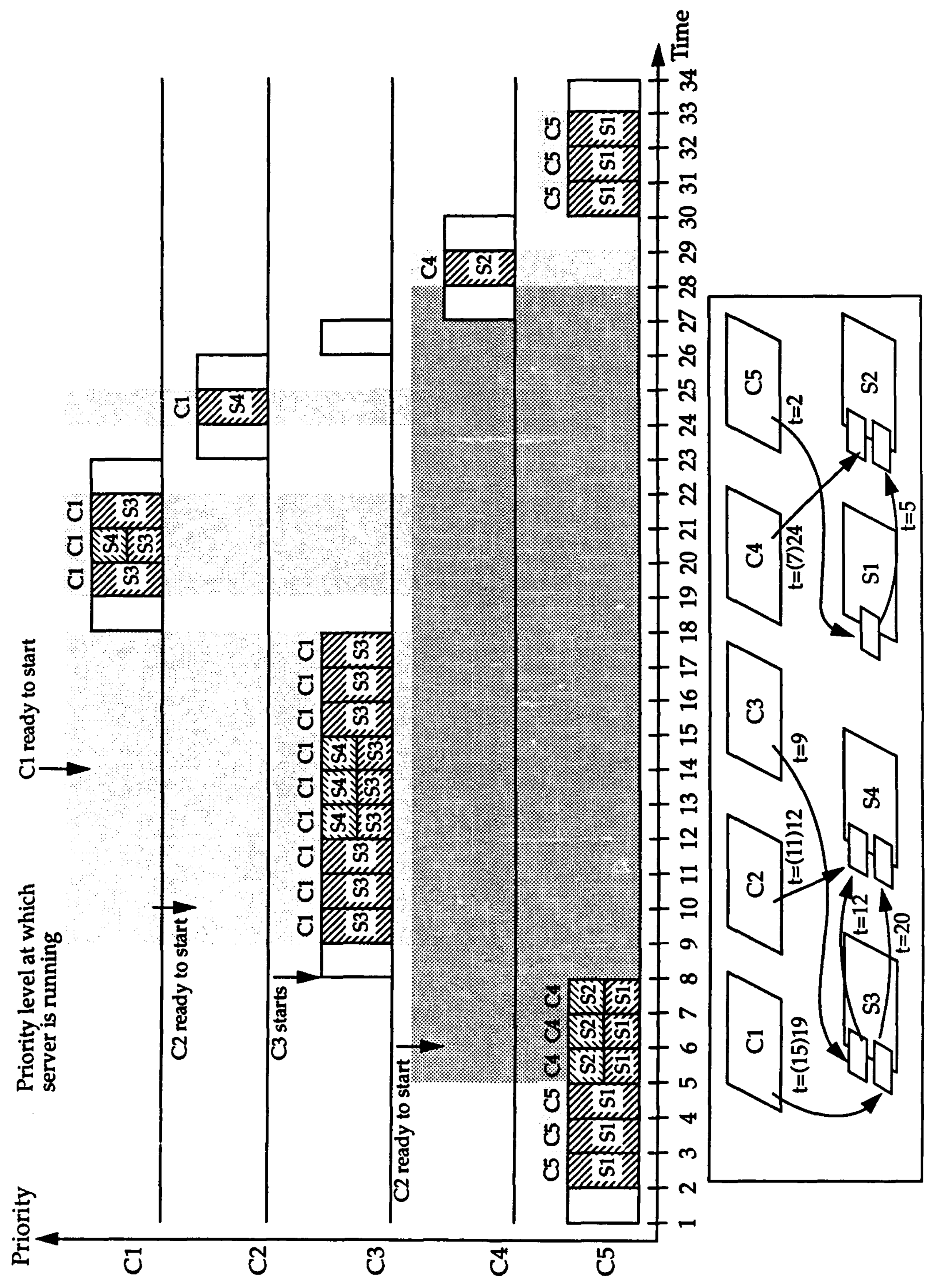
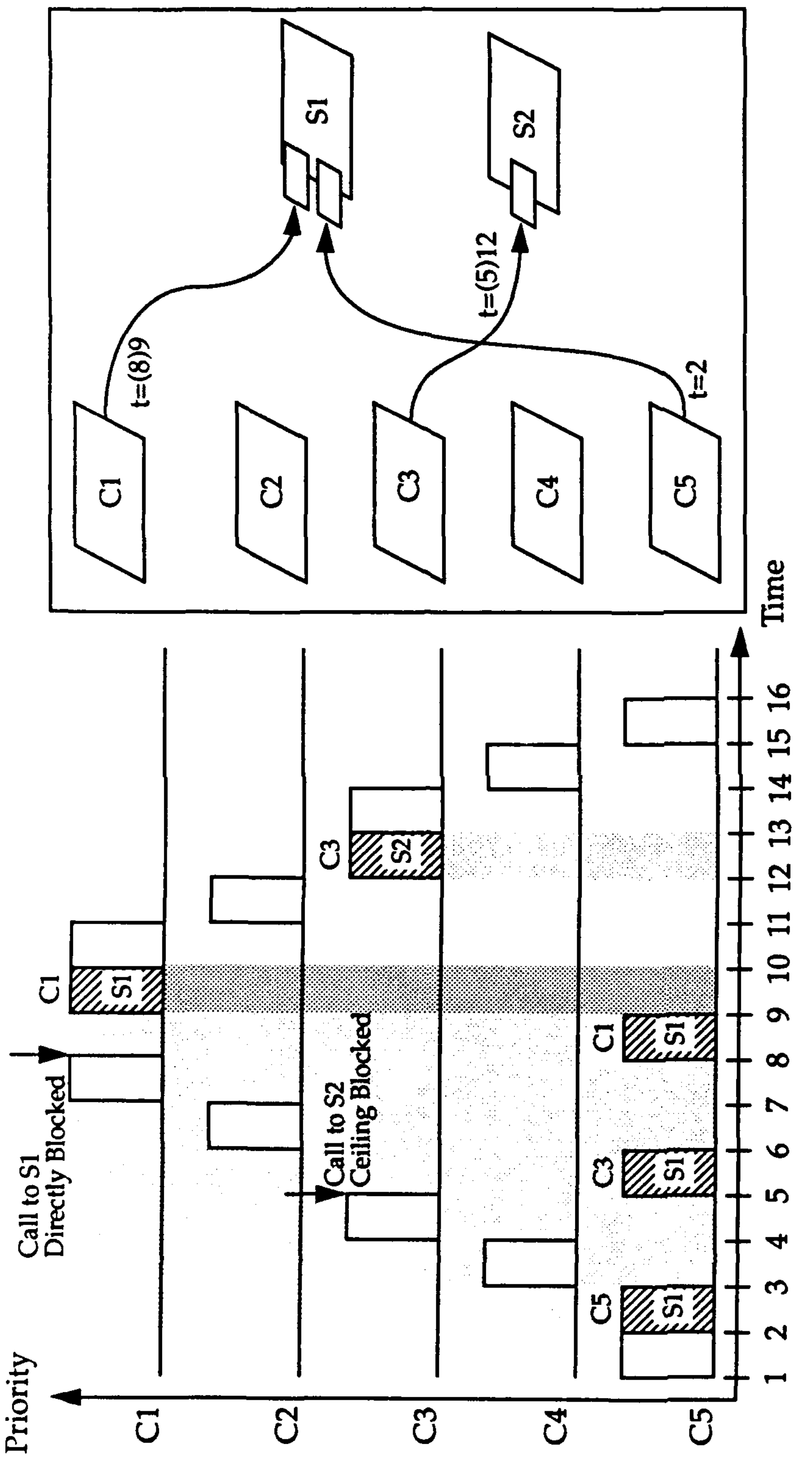


Figure A-32 PC_12: Events When Servers Execute at Ceiling Priority



t=1 : C5 begins to execute
t=2 : C5 rendezvous with S1
t=3 : C4 preempts S1
t=4 : C3 preempts C4
t=5 : C3 attempts to call S2 [CEILING BLOCKED]
S1 resumes at C3's priority on behalf of C5
t=6 : C2 preempts S1
t=7 : C1 preempts C2
t=8 : C1 attempts to call S1 [DIRECTLY BLOCKED]
S1 resumes at C1's priority on behalf of C5
t=9 : S1 completes
C1 preempts C5 and rendezvous with S1
t=10: S1 completes and C1 resumes
t=11: C1 completes and C2 resumes
t=12: C2 completes
C3 resumes and rendezvous with S2
t=13: S2 completes and C3 resumes
t=14: C3 completes and C4 resumes
t=15: C4 completes and C5 resumes
t=16: C5 completes

Figure A-33 PC_13: Priority of Ceiling-Blocked Task Is Inherited

Appendix B Logical Test Suite Expected Events

```
[Task: C2 Begins execution                              at t = 1]
[Task: C1 Begins execution                              at t = 2]
[Task: C1 Ends execution                                at t = 5]
[Task: C2 Ends execution                                at t = 7]
******************* Test Complete *******************
```

Figure B-1 Expected Events for PS_01

```
[Task: C1 Begins execution                                    at t = 1]
[Task: C1 Ends execution                                      at t = 4]
[Task: C2 Begins execution                                    at t = 4]
[Task: C2 Ends execution                                      at t = 5]
[Task: C3 Begins execution                                    at t = 5]
[Task: C3 Ends execution                                      at t = 6]
******************** Test Complete ********************
```

Figure B-2 Expected Events for PS_02

```
[Task: C1 Begins execution                                        at t = 1]
[Task: C1 Ends execution                                          at t = 4]
[Task: C2 Begins execution                                        at t = 4]
[Task: C2 Ends execution                                          at t = 5]
[Task: C3 Begins execution                                        at t = 5]
[Task: C3 Ends execution                                          at t = 6]
[Task: C4 Begins execution                                        at t = 6]
[Task: C4 Ends execution                                          at t = 7]
[Task: C5 Begins execution                                        at t = 7]
[Task: C5 Ends execution                                          at t = 8]
******************* Test Complete *******************
```

Figure B-3 Expected Events for PS_03

```
[Task: C3 Calls server:                  S1      at t = 1]
[Task: S1 Begins execution on behalf of: C3      at t = 1]
[Task: C1 Begins execution                       at t = 2]
[Task: C1 Ends execution                         at t = 3]
[Task: C1 Calls server:                  S1      at t = 3]
[Task: S1 Ends execution on behalf of:   C3      at t = 5]
[Task: S1 Begins execution on behalf of: C1      at t = 5]
[Task: S1 Ends execution on behalf of:   C1      at t = 6]
[Task: C1 Begins execution                       at t = 6]
[Task: C1 Ends execution                         at t = 7]
[Task: C2 Begins execution                       at t = 7]
[Task: C2 Ends execution                         at t = 8]
[Task: C3 Begins execution                       at t = 8]
[Task: C3 Ends execution                         at t = 9]
****************** Test Complete ******************
```

Figure B-4 Expected Events for BI_01

```
[Task: C5 Calls server:                     S1      at t = 1]
[Task: S1 Begins execution on behalf of:    C5      at t = 1]
[Task: C4 Begins execution                          at t = 2]
[Task: C3 Begins execution                          at t = 3]
[Task: C3 Ends execution                            at t = 4]
[Task: C3 Calls server:                     S1      at t = 4]
[Task: C2 Begins execution                          at t = 5]
[Task: C1 Begins execution                          at t = 6]
[Task: C1 Ends execution                            at t = 7]
[Task: C1 Calls server:                     S1      at t = 7]
[Task: S1 Ends execution on behalf of:      C5      at t = 8]
[Task: S1 Begins execution on behalf of:    C1      at t = 8]
[Task: S1 Ends execution on behalf of:      C1      at t = 9]
[Task: C1 Begins execution                          at t = 9]
[Task: C1 Ends execution                            at t = 10]
[Task: C2 Ends execution                            at t = 11]
[Task: S1 Begins execution on behalf of:    C3      at t = 11]
[Task: S1 Ends execution on behalf of:      C3      at t = 12]
[Task: C3 Begins execution                          at t = 12]
[Task: C3 Ends execution                            at t = 13]
[Task: C4 Ends execution                            at t = 14]
[Task: C5 Begins execution                          at t = 14]
[Task: C5 Ends execution                            at t = 15]
******************* Test Complete *******************
```

Figure B-5 Expected Events for BI_02

```
[Task: C1 Calls server:                      S1        at t = 1]
[Task: S1 Begins execution on behalf of:     C1        at t = 1]
[Task: S1 Ends execution on behalf of:       C1        at t = 3]
[Task: S1 Calls server:                      S2        at t = 3]
[Task: S2 Begins execution on behalf of:     S1        at t = 3]
[Task: S2 Ends execution on behalf of:       S1        at t = 5]
[Task: S1 Begins execution on behalf of:     C1        at t = 5]
[Task: S1 Ends execution on behalf of:       C1        at t = 7]
[Task: C1 Begins execution                             at t = 7]
[Task: C1 Ends execution                               at t = 8]
[Task: C2 Begins execution                             at t = 8]
[Task: C2 Ends execution                               at t = 9]
******************** Test Complete ********************
```

Figure B-6 Expected Events for BI_03

```
[Task: C2 Calls server:                      S1        at t = 1]
[Task: S1 Begins execution on behalf of:     C2        at t = 1]
[Task: S1 Ends execution on behalf of:       C2        at t = 4]
[Task: S1 Calls server:                      S2        at t = 4]
[Task: S2 Begins execution on behalf of:     S1        at t = 4]
[Task: C1 Begins execution                             at t = 6]
[Task: C1 Ends execution                               at t = 7]
[Task: S2 Ends execution on behalf of:       S1        at t = 8]
[Task: S1 Begins execution on behalf of:     C2        at t = 8]
[Task: S1 Ends execution on behalf of:       C2        at t = 11]
[Task: C2 Begins execution                             at t = 11]
[Task: C2 Ends execution                               at t = 12]
[Task: C3 Begins execution                             at t = 12]
[Task: C3 Ends execution                               at t = 13]
******************** Test Complete ********************
```

Figure B-7 Expected Events for BI_04

```
[Task: C3 Begins execution                                at t = 1]
[Task: C3 Ends execution                                  at t = 2]
[Task: C3 Calls server:                    S1             at t = 2]
[Task: S1 Begins execution on behalf of:   C3             at t = 2]
[Task: S1 Ends execution on behalf of:     C3             at t = 4]
[Task: S1 Calls server:                    S2             at t = 4]
[Task: S2 Begins execution on behalf of:   S1             at t = 4]
[Task: C2 Begins execution                                at t = 5]
[Task: C1 Begins execution                                at t = 6]
[Task: C1 Ends execution                                  at t = 7]
[Task: C1 Calls server:                    S1             at t = 7]
[Task: S2 Ends execution on behalf of:     S1             at t = 8]
[Task: S1 Begins execution on behalf of:   C3             at t = 8]
[Task: S1 Ends execution on behalf of:     C3             at t = 10]
[Task: S1 Begins execution on behalf of:   C1             at t = 10]
[Task: S1 Ends execution on behalf of:     C1             at t = 11]
[Task: C1 Begins execution                                at t = 11]
[Task: C1 Ends execution                                  at t = 12]
[Task: C2 Ends execution                                  at t = 13]
[Task: C3 Begins execution                                at t = 13]
[Task: C3 Ends execution                                  at t = 14]
******************* Test Complete *******************
```

Figure B-8 Expected Events for BI_05

```
[Task: C2 Begins execution                              at t = 1]
[Task: C1 Begins execution                              at t = 2]
[Task: C1 Ends execution                                at t = 3]
[Task: C1 Calls server:                    S1           at t = 3]
[Task: S1 Begins execution on behalf of:   C1           at t = 3]
[Task: S1 Ends execution on behalf of:     C1           at t = 4]
[Task: S1 Begins Suspension on behalf of:  C1           at t = 4]
[Task: C2 Ends execution                                at t = 5]
[Task: S1 Ends Suspension on behalf of:    C1           at t = 6]
[Task: S1 Begins execution on behalf of:   C1           at t = 6]
[Task: S1 Ends execution on behalf of:     C1           at t = 7]
[Task: C1 Begins execution                              at t = 7]
[Task: C1 Ends execution                                at t = 8]
******************** Test Complete ********************
```

Figure B-9 Expected Events for BI_06

```
[Task: C1 Begins execution                          at t = 1]
[Task: C1 Ends execution                            at t = 2]
[Task: C1 Calls server:                    S1       at t = 2]
[Task: S1 Begins execution on behalf of:   C1       at t = 2]
[Task: S1 Ends execution on behalf of:     C1       at t = 3]
[Task: S1 Begins Suspension on behalf of:  C1       at t = 3]
[Task: C2 Begins execution                          at t = 3]
[Task: C2 Ends execution                            at t = 4]
[Task: C2 Calls server:                    S2       at t = 4]
[Task: S2 Begins execution on behalf of:   C2       at t = 4]
[Task: S1 Ends Suspension on behalf of:    C1       at t = 5]
[Task: S1 Begins execution on behalf of:   C1       at t = 5]
[Task: S1 Ends execution on behalf of:     C1       at t = 6]
[Task: C1 Begins execution                          at t = 6]
[Task: C1 Ends execution                            at t = 7]
[Task: S2 Ends execution on behalf of:     C2       at t = 8]
[Task: C2 Begins execution                          at t = 8]
[Task: C2 Ends execution                            at t = 9]
******************** Test Complete ********************
```

Figure B-10 Expected Events for BI_07

```
[Task: C1 Begins execution                              at t = 1]
[Task: C1 Ends execution                                at t = 2]
[Task: C1 Calls server:                     S1          at t = 2]
[Task: S1 Begins execution on behalf of:    C1          at t = 2]
[Task: S1 Ends execution on behalf of:      C1          at t = 3]
[Task: C1 Begins execution                              at t = 3]
[Task: C1 Ends execution                                at t = 4]
[Task: C2 Begins execution                              at t = 4]
[Task: C2 Ends execution                                at t = 5]
[Task: C2 Calls server:                     S1          at t = 5]
[Task: S1 Begins execution on behalf of:    C2          at t = 5]
[Task: S1 Ends execution on behalf of:      C2          at t = 6]
[Task: C2 Begins execution                              at t = 6]
[Task: C2 Ends execution                                at t = 7]
******************* Test Complete *******************
```

Figure B-11 Expected Events for BI_08

```
[Task: C5 Begins execution                          at t = 1]
[Task: C5 Ends execution                            at t = 2]
[Task: C5 Calls server:                     S1      at t = 2]
[Task: S1 Begins execution on behalf of:    C5      at t = 2]
[Task: C4 Begins execution                          at t = 3]
[Task: C4 Ends execution                            at t = 4]
[Task: C4 Calls server:                     S2      at t = 4]
[Task: S2 Begins execution on behalf of:    C4      at t = 4]
[Task: C3 Begins execution                          at t = 5]
[Task: C2 Begins execution                          at t = 6]
[Task: C2 Ends execution                            at t = 7]
[Task: C2 Calls server:                     S1      at t = 7]
[Task: C1 Begins execution                          at t = 8]
[Task: C1 Ends execution                            at t = 9]
[Task: C1 Calls server:                     S2      at t = 9]
[Task: S2 Ends execution on behalf of:      C4      at t = 10]
[Task: S2 Calls server:                     S1      at t = 10]
[Task: S1 Ends execution on behalf of:      C5      at t = 12]
[Task: S1 Begins execution on behalf of:    S2      at t = 12]
[Task: S1 Ends execution on behalf of:      S2      at t = 14]
[Task: S2 Begins execution on behalf of:    C4      at t = 14]
[Task: S2 Ends execution on behalf of:      C4      at t = 16]
[Task: S2 Begins execution on behalf of:    C1      at t = 16]
[Task: S2 Ends execution on behalf of:      C1      at t = 17]
[Task: C1 Begins execution                          at t = 17]
[Task: C1 Ends execution                            at t = 18]
[Task: S1 Begins execution on behalf of:    C2      at t = 18]
[Task: S1 Ends execution on behalf of:      C2      at t = 19]
[Task: C2 Begins execution                          at t = 19]
[Task: C2 Ends execution                            at t = 20]
[Task: C3 Ends execution                            at t = 21]
[Task: C4 Begins execution                          at t = 21]
[Task: C4 Ends execution                            at t = 22]
[Task: C5 Begins execution                          at t = 22]
[Task: C5 Ends execution                            at t = 23]
****************** Test Complete ******************
```

Figure B-12 Expected Events for BI_09

```
[Task: C5 Begins execution                              at t = 1]
[Task: C5 Ends execution                                at t = 2]
[Task: C5 Calls server:                      S1         at t = 2]
[Task: S1 Begins execution on behalf of:     C5         at t = 2]
[Task: S1 Ends execution on behalf of:       C5         at t = 5]
[Task: S1 Calls server:                      S2         at t = 5]
[Task: S2 Begins execution on behalf of:     S1         at t = 5]
[Task: C4 Begins execution                              at t = 6]
[Task: C4 Ends execution                                at t = 7]
[Task: C4 Calls server:                      S2         at t = 7]
[Task: C3 Begins execution                              at t = 8]
[Task: C3 Ends execution                                at t = 9]
[Task: C3 Calls server:                      S3         at t = 9]
[Task: S3 Begins execution on behalf of:     C3         at t = 9]
[Task: C2 Begins execution                              at t = 10]
[Task: C2 Ends execution                                at t = 11]
[Task: C2 Calls server:                      S4         at t = 11]
[Task: S4 Begins execution on behalf of:     C2         at t = 11]
[Task: S4 Ends execution on behalf of:       C2         at t = 12]
[Task: C2 Begins execution                              at t = 12]
[Task: C2 Ends execution                                at t = 13]
[Task: C1 Begins execution                              at t = 14]
[Task: C1 Ends execution                                at t = 15]
[Task: C1 Calls server:                      S3         at t = 15]
[Task: S3 Ends execution on behalf of:       C3         at t = 16]
[Task: S3 Calls server:                      S4         at t = 16]
[Task: S4 Begins execution on behalf of:     S3         at t = 16]
[Task: S4 Ends execution on behalf of:       S3         at t = 19]
[Task: S3 Begins execution on behalf of:     C3         at t = 19]
[Task: S3 Ends execution on behalf of:       C3         at t = 22]
[Task: S3 Begins execution on behalf of:     C1         at t = 22]
[Task: S3 Ends execution on behalf of:       C1         at t = 23]
[Task: S3 Calls server:                      S4         at t = 23]
[Task: S4 Begins execution on behalf of:     S3         at t = 23]
[Task: S4 Ends execution on behalf of:       S3         at t = 24]
[Task: S3 Begins execution on behalf of:     C1         at t = 24]
[Task: S3 Ends execution on behalf of:       C1         at t = 25]
[Task: C1 Begins execution                              at t = 25]
[Task: C1 Ends execution                                at t = 26]
[Task: C3 Begins execution                              at t = 26]
[Task: C3 Ends execution                                at t = 27]
[Task: S2 Ends execution on behalf of:       S1         at t = 28]
[Task: S2 Begins execution on behalf of:     C4         at t = 28]
[Task: S2 Ends execution on behalf of:       C4         at t = 29]
[Task: C4 Begins execution                              at t = 29]
[Task: C4 Ends execution                                at t = 30]
[Task: S1 Begins execution on behalf of:     C5         at t = 30]
[Task: S1 Ends execution on behalf of:       C5         at t = 33]
[Task: C5 Begins execution                              at t = 33]
[Task: C5 Ends execution                                at t = 34]
****************** Test Complete ******************
```

Figure B-13 Expected Events for BI_10

```
[Task: C2 Calls server:                          S2        at t = 1]
[Task: S2 Begins execution on behalf of:         C2        at t = 1]
[Task: C1 Begins execution                                 at t = 2]
[Task: C1 Ends execution                                   at t = 3]
[Task: C1 Calls server:                          S1        at t = 3]
[Task: S2 Ends execution on behalf of:           C2        at t = 4]
[Task: S1 Begins execution on behalf of:         C1        at t = 4]
[Task: S1 Ends execution on behalf of:           C1        at t = 5]
[Task: S1 Calls server:                          S2        at t = 5]
[Task: S2 Begins execution on behalf of:         S1        at t = 5]
[Task: S2 Ends execution on behalf of:           S1        at t = 6]
[Task: S1 Begins execution on behalf of:         C1        at t = 6]
[Task: S1 Ends execution on behalf of:           C1        at t = 7]
[Task: C1 Begins execution                                 at t = 7]
[Task: C1 Ends execution                                   at t = 8]
[Task: C2 Begins execution                                 at t = 8]
[Task: C2 Ends execution                                   at t = 9]
******************* Test Complete *******************
```

Figure B-14 Expected Events for PC_01

```
[Task: C2 Begins execution                                 at t = 1]
[Task: C2 Ends execution                                   at t = 2]
[Task: C2 Calls server:                          S2        at t = 2]
[Task: S2 Begins execution on behalf of:         C2        at t = 2]
[Task: C1 Begins execution                                 at t = 3]
[Task: C1 Ends execution                                   at t = 4]
[Task: C1 Calls server:                          S1        at t = 4]
[Task: S2 Ends execution on behalf of:           C2        at t = 5]
[Task: S2 Calls server:                          S1        at t = 5]
[Task: S1 Begins execution on behalf of:         S2        at t = 5]
[Task: S1 Ends execution on behalf of:           S2        at t = 7]
[Task: S2 Begins execution on behalf of:         C2        at t = 7]
[Task: S2 Ends execution on behalf of:           C2        at t = 9]
[Task: S1 Begins execution on behalf of:         C1        at t = 9]
[Task: S1 Ends execution on behalf of:           C1        at t = 11]
[Task: S1 Calls server:                          S2        at t = 11]
[Task: S2 Begins execution on behalf of:         S1        at t = 11]
[Task: S2 Ends execution on behalf of:           S1        at t = 13]
[Task: S1 Begins execution on behalf of:         C1        at t = 13]
[Task: S1 Ends execution on behalf of:           C1        at t = 15]
[Task: C1 Begins execution                                 at t = 15]
[Task: C1 Ends execution                                   at t = 16]
[Task: C2 Begins execution                                 at t = 16]
[Task: C2 Ends execution                                   at t = 17]
****************** Test Complete ******************
```

Figure B-15 Expected Events for PC_02

```
[Task: C3 Calls server:                        S1      at t = 1]
[Task: S1 Begins execution on behalf of:       C3      at t = 1]
[Task: C2 Begins execution                             at t = 2]
[Task: C2 Ends execution                               at t = 3]
[Task: C2 Calls server:                        S2      at t = 3]
[Task: C1 Begins execution                             at t = 4]
[Task: C1 Ends execution                               at t = 5]
[Task: C1 Calls server:                        S1      at t = 5]
[Task: S1 Ends execution on behalf of:         C3      at t = 6]
[Task: S1 Calls server:                        S2      at t = 6]
[Task: S2 Begins execution on behalf of:       S1      at t = 6]
[Task: S2 Ends execution on behalf of:         S1      at t = 9]
[Task: S1 Begins execution on behalf of:       C3      at t = 9]
[Task: S1 Ends execution on behalf of:         C3      at t = 12]
[Task: S1 Begins execution on behalf of:       C1      at t = 12]
[Task: S1 Ends execution on behalf of:         C1      at t = 13]
[Task: S1 Calls server:                        S2      at t = 13]
[Task: S2 Begins execution on behalf of:       S1      at t = 13]
[Task: S2 Ends execution on behalf of:         S1      at t = 14]
[Task: S1 Begins execution on behalf of:       C1      at t = 14]
[Task: S1 Ends execution on behalf of:         C1      at t = 15]
[Task: C1 Begins execution                             at t = 15]
[Task: C1 Ends execution                               at t = 16]
[Task: S2 Begins execution on behalf of:       C2      at t = 16]
[Task: S2 Ends execution on behalf of:         C2      at t = 17]
[Task: S2 Calls server:                        S1      at t = 17]
[Task: S1 Begins execution on behalf of:       S2      at t = 17]
[Task: S1 Ends execution on behalf of:         S2      at t = 18]
[Task: S2 Begins execution on behalf of:       C2      at t = 18]
[Task: S2 Ends execution on behalf of:         C2      at t = 19]
[Task: C2 Begins execution                             at t = 19]
[Task: C2 Ends execution                               at t = 20]
[Task: C3 Begins execution                             at t = 20]
[Task: C3 Ends execution                               at t = 21]
****************** Test Complete ******************
```

Figure B-17 Expected Events for PC_04

```
[Task: C2 Begins execution                                  at t = 1]
[Task: C2 Ends execution                                    at t = 2]
[Task: C2 Calls server:                      S2             at t = 2]
[Task: S2 Begins execution on behalf of:     C2             at t = 2]
[Task: C1 Begins execution                                  at t = 3]
[Task: C1 Ends execution                                    at t = 4]
[Task: C1 Calls server:                      S1             at t = 4]
[Task: S1 Begins execution on behalf of:     C1             at t = 4]
[Task: S1 Ends execution on behalf of:       C1             at t = 5]
[Task: C1 Begins execution                                  at t = 5]
[Task: C1 Ends execution                                    at t = 6]
[Task: S2 Ends execution on behalf of:       C2             at t = 7]
[Task: C2 Begins execution                                  at t = 7]
[Task: C2 Ends execution                                    at t = 8]
******************** Test Complete ********************
```

Figure B-18 Expected Events for PC_05

```
[Task: C3 Calls server:                    S2      at t = 1]
[Task: S2 Begins execution on behalf of:   C3      at t = 1]
[Task: C2 Begins execution                         at t = 2]
[Task: C2 Ends execution                           at t = 3]
[Task: C2 Calls server:                    S2      at t = 3]
[Task: C1 Begins execution                         at t = 4]
[Task: C1 Ends execution                           at t = 5]
[Task: C1 Calls server:                    S1      at t = 5]
[Task: S1 Begins execution on behalf of:   C1      at t = 5]
[Task: S1 Ends execution on behalf of:     C1      at t = 6]
[Task: C1 Begins execution                         at t = 6]
[Task: C1 Ends execution                           at t = 7]
[Task: S2 Ends execution on behalf of:     C3      at t = 8]
[Task: S2 Begins execution on behalf of:   C2      at t = 8]
[Task: S2 Ends execution on behalf of:     C2      at t = 9]
[Task: C2 Begins execution                         at t = 9]
[Task: C2 Ends execution                           at t = 10]
[Task: C3 Begins execution                         at t = 10]
[Task: C3 Ends execution                           at t = 11]
******************* Test Complete *******************
```

Figure B-19 Expected Events for PC_06

```
[Task: C1 Begins execution                              at t = 1]
[Task: C1 Ends execution                                at t = 2]
[Task: C1 Calls server:                      S1         at t = 2]
[Task: S1 Begins execution on behalf of:     C1         at t = 2]
[Task: S1 Ends execution on behalf of:       C1         at t = 3]
[Task: S1 Begins Suspension on behalf of:    C1         at t = 3]
[Task: C2 Begins execution                              at t = 3]
[Task: C2 Ends execution                                at t = 4]
[Task: C2 Calls server:                      S2         at t = 4]
[Task: S1 Ends Suspension on behalf of:      C1         at t = 5]
[Task: S1 Begins execution on behalf of:     C1         at t = 5]
[Task: S1 Ends execution on behalf of:       C1         at t = 6]
[Task: C1 Begins execution                              at t = 6]
[Task: C1 Ends execution                                at t = 7]
[Task: S2 Begins execution on behalf of:     C2         at t = 7]
[Task: S2 Ends execution on behalf of:       C2         at t = 9]
[Task: C2 Begins execution                              at t = 9]
[Task: C2 Ends execution                                at t = 10]
******************* Test Complete *******************
```

Figure B-20 Expected Events for PC_07

```
[Task: C3 Calls server:                        S3      at t = 1]
[Task: S3 Begins execution on behalf of:       C3      at t = 1]
[Task: C2 Begins execution                             at t = 2]
[Task: C2 Ends execution                               at t = 3]
[Task: C2 Calls server:                        S2      at t = 3]
[Task: C1 Begins execution                             at t = 4]
[Task: C1 Ends execution                               at t = 5]
[Task: C1 Calls server:                        S1      at t = 5]
[Task: S1 Begins execution on behalf of:       C1      at t = 5]
[Task: S1 Ends execution on behalf of:         C1      at t = 6]
[Task: S1 Calls server:                        S2      at t = 6]
[Task: S2 Begins execution on behalf of:       S1      at t = 6]
[Task: S2 Ends execution on behalf of:         S1      at t = 7]
[Task: S1 Begins execution on behalf of:       C1      at t = 7]
[Task: S1 Ends execution on behalf of:         C1      at t = 8]
[Task: C1 Begins execution                             at t = 8]
[Task: C1 Ends execution                               at t = 9]
[Task: S3 Ends execution on behalf of:         C3      at t = 10]
[Task: S3 Calls server:                        S1      at t = 10]
[Task: S1 Begins execution on behalf of:       S3      at t = 10]
[Task: S1 Ends execution on behalf of:         S3      at t = 13]
[Task: S3 Begins execution on behalf of:       C3      at t = 13]
[Task: S3 Ends execution on behalf of:         C3      at t = 16]
[Task: S2 Begins execution on behalf of:       C2      at t = 16]
[Task: S2 Ends execution on behalf of:         C2      at t = 17]
[Task: S2 Calls server:                        S3      at t = 17]
[Task: S3 Begins execution on behalf of:       S2      at t = 17]
[Task: S3 Ends execution on behalf of:         S2      at t = 18]
[Task: S2 Begins execution on behalf of:       C2      at t = 18]
[Task: S2 Ends execution on behalf of:         C2      at t = 19]
[Task: C2 Begins execution                             at t = 19]
[Task: C2 Ends execution                               at t = 20]
[Task: C3 Begins execution                             at t = 20]
[Task: C3 Ends execution                               at t = 21]
******************* Test Complete *******************
```

Figure B-21 Expected Events for PC_08

```
[Task: C3 Calls server:                      S3        at t = 1]
[Task: S3 Begins execution on behalf of:     C3        at t = 1]
[Task: C2 Begins execution                             at t = 2]
[Task: C2 Ends execution                               at t = 3]
[Task: C2 Calls server:                      S2        at t = 3]
[Task: S3 Ends execution on behalf of:       C3        at t = 4]
[Task: S3 Calls server:                      S1        at t = 4]
[Task: S1 Begins execution on behalf of:     S3        at t = 4]
[Task: C1 Begins execution                             at t = 5]
[Task: C1 Ends execution                               at t = 6]
[Task: C1 Calls server: S1                             at t = 6]
[Task: S1 Ends execution on behalf of:       S3        at t = 7]
[Task: S1 Begins execution on behalf of:     C1        at t = 7]
[Task: S1 Ends execution on behalf of:       C1        at t = 8]
[Task: S1 Calls server:                      S2        at t = 8]
[Task: S2 Begins execution on behalf of:     S1        at t = 8]
[Task: S2 Ends execution on behalf of:       S1        at t = 9]
[Task: S1 Begins execution on behalf of:     C1        at t = 9]
[Task: S1 Ends execution on behalf of:       C1        at t = 10]
[Task: C1 Begins execution                             at t = 10]
[Task: C1 Ends execution                               at t = 11]
[Task: S3 Begins execution on behalf of:     C3        at t = 11]
[Task: S3 Ends execution on behalf of:       C3        at t = 13]
[Task: S2 Begins execution on behalf of:     C2        at t = 13]
[Task: S2 Ends execution on behalf of:       C2        at t = 15]
[Task: S2 Calls server:                      S3        at t = 15]
[Task: S3 Begins execution on behalf of:     S2        at t = 15]
[Task: S3 Ends execution on behalf of:       S2        at t = 17]
[Task: S2 Begins execution on behalf of:     C2        at t = 17]
[Task: S2 Ends execution on behalf of:       C2        at t = 19]
[Task: C2 Begins execution                             at t = 19]
[Task: C2 Ends execution                               at t = 20]
[Task: C3 Begins execution                             at t = 20]
[Task: C3 Ends execution                               at t = 21]
******************* Test Complete *******************
```

Figure B-22 Expected Events for PC_09

```
[Task: C2 Calls server:                        S2       at t = 1]
[Task: S2 Begins execution on behalf of:       C2       at t = 1]
[Task: C1 Begins execution                              at t = 2]
[Task: C1 Ends execution                                at t = 3]
[Task: C1 Calls server:                        S1       at t = 3]
[Task: S2 Ends execution on behalf of:         C2       at t = 4]
[Task: S2 Begins Suspension on behalf of:      C2       at t = 4]
[Task: S2 Ends Suspension on behalf of:        C2       at t = 6]
[Task: S2 Begins execution on behalf of:       C2       at t = 6]
[Task: S2 Ends execution on behalf of:         C2       at t = 8]
[Task: S1 Begins execution on behalf of:       C1       at t = 8]
[Task: S1 Ends execution on behalf of:         C1       at t = 9]
[Task: S1 Calls server:                        S2       at t = 9]
[Task: S2 Begins execution on behalf of:       S1       at t = 9]
[Task: S2 Ends execution on behalf of:         S1       at t = 10]
[Task: S2 Begins Suspension on behalf of:      S1       at t = 10]
[Task: C2 Begins execution                              at t = 10]
[Task: C2 Ends execution                                at t = 11]
[Task: S2 Ends Suspension on behalf of:        S1       at t = 12]
[Task: S2 Begins execution on behalf of:       S1       at t = 12]
[Task: S2 Ends execution on behalf of:         S1       at t = 13]
[Task: S1 Begins execution on behalf of:       C1       at t = 13]
[Task: S1 Ends execution on behalf of:         C1       at t = 14]
[Task: C1 Begins execution                              at t = 14]
[Task: C1 Ends execution                                at t = 15]
******************** Test Complete ********************
```

Figure B-23 Expected Events for PC_10

```
[Task: C5 Begins execution                          at t = 1]
[Task: C5 Ends execution                            at t = 2]
[Task: C5 Calls server:                     S1      at t = 2]
[Task: S1 Begins execution on behalf of:    C5      at t = 2]
[Task: C4 Begins execution                          at t = 3]
[Task: C4 Ends execution                            at t = 4]
[Task: C4 Calls server:                     S2      at t = 4]
[Task: C3 Begins execution                          at t = 5]
[Task: C2 Begins execution                          at t = 6]
[Task: C2 Ends execution                            at t = 7]
[Task: C2 Calls server:                     S1      at t = 7]
[Task: C1 Begins execution                          at t = 8]
[Task: C1 Ends execution                            at t = 9]
[Task: C1 Calls server:                     S2      at t = 9]
[Task: S2 Begins execution on behalf of:    C1      at t = 9]
[Task: S2 Ends execution on behalf of:      C1      at t = 10]
[Task: C1 Begins execution                          at t = 10]
[Task: C1 Ends execution                            at t = 11]
[Task: S1 Ends execution on behalf of:      C5      at t = 12]
[Task: S1 Begins execution on behalf of:    C2      at t = 12]
[Task: S1 Ends execution on behalf of:      C2      at t = 13]
[Task: C2 Begins execution                          at t = 13]
[Task: C2 Ends execution                            at t = 14]
[Task: C3 Ends execution                            at t = 15]
[Task: S2 Begins execution on behalf of:    C4      at t = 15]
[Task: S2 Ends execution on behalf of:      C4      at t = 17]
[Task: S2 Calls server:                     S1      at t = 17]
[Task: S1 Begins execution on behalf of:    S2      at t = 17]
[Task: S1 Ends execution on behalf of:      S2      at t = 19]
[Task: S2 Begins execution on behalf of:    C4      at t = 19]
[Task: S2 Ends execution on behalf of:      C4      at t = 21]
[Task: C4 Begins execution                          at t = 21]
[Task: C4 Ends execution                            at t = 22]
[Task: C5 Begins execution                          at t = 22]
[Task: C5 Ends execution                            at t = 23]
****************** Test Complete ******************
```

Figure B-24 Expected Events for PC_11

```
[Task: C5 Begins execution                            at t = 1]
[Task: C5 Ends execution                              at t = 2]
[Task: C5 Calls server:                    S1         at t = 2]
[Task: S1 Begins execution on behalf of:   C5         at t = 2]
[Task: S1 Ends execution on behalf of:     C5         at t = 5]
[Task: S1 Calls server:                    S2         at t = 5]
[Task: S2 Begins execution on behalf of:   S1         at t = 5]
[Task: C4 Begins execution                            at t = 6]
[Task: C4 Ends execution                              at t = 7]
[Task: C4 Calls server:                    S2         at t = 7]
[Task: C3 Begins execution                            at t = 8]
[Task: C3 Ends execution                              at t = 9]
[Task: C3 Calls server:                    S3         at t = 9]
[Task: S3 Begins execution on behalf of:   C3         at t = 9]
[Task: C2 Begins execution                            at t = 10]
[Task: C2 Ends execution                              at t = 11]
[Task: C2 Calls server:                    S4         at t = 11]
[Task: S3 Ends execution on behalf of:     C3         at t = 13]
[Task: S3 Calls server:                    S4         at t = 13]
[Task: S4 Begins execution on behalf of:   S3         at t = 13]
[Task: C1 Begins execution                            at t = 14]
[Task: C1 Ends execution                              at t = 15]
[Task: C1 Calls server:                    S3         at t = 15]
[Task: S4 Ends execution on behalf of:     S3         at t = 17]
[Task: S3 Begins execution on behalf of:   C3         at t = 17]
[Task: S3 Ends execution on behalf of:     C3         at t = 20]
[Task: S3 Begins execution on behalf of:   C1         at t = 20]
[Task: S3 Ends execution on behalf of:     C1         at t = 21]
[Task: S3 Calls server:                    S4         at t = 21]
[Task: S4 Begins execution on behalf of:   S3         at t = 21]
[Task: S4 Ends execution on behalf of:     S3         at t = 22]
[Task: S3 Begins execution on behalf of:   C1         at t = 22]
[Task: S3 Ends execution on behalf of:     C1         at t = 23]
[Task: C1 Begins execution                            at t = 23]
[Task: C1 Ends execution                              at t = 24]
[Task: S4 Begins execution on behalf of:   C2         at t = 24]
[Task: S4 Ends execution on behalf of:     C2         at t = 25]
[Task: C2 Begins execution                            at t = 25]
[Task: C2 Ends execution                              at t = 26]
[Task: C3 Begins execution                            at t = 26]
[Task: C3 Ends execution                              at t = 27]
[Task: S2 Ends execution on behalf of:     S1         at t = 28]
[Task: S2 Begins execution on behalf of:   C4         at t = 28]
[Task: S2 Ends execution on behalf of:     C4         at t = 29]
[Task: C4 Begins execution                            at t = 29]
[Task: C4 Ends execution                              at t = 30]
[Task: S1 Begins execution on behalf of:   C5         at t = 30]
[Task: S1 Ends execution on behalf of:     C5         at t = 33]
[Task: C5 Begins execution                            at t = 33]
[Task: C5 Ends execution                              at t = 34]
****************** Test Complete ******************
```

Figure B-25 Expected Events for PC_12

```
[Task: C5 Begins execution                          at t = 1]
[Task: C5 Ends execution                            at t = 2]
[Task: C5 Calls server:                    S1       at t = 2]
[Task: S1 Begins execution on behalf of:   C5       at t = 2]
[Task: C4 Begins execution                          at t = 3]
[Task: C3 Begins execution                          at t = 4]
[Task: C3 Ends execution                            at t = 5]
[Task: C3 Calls server:                    S2       at t = 5]
[Task: C2 Begins execution                          at t = 6]
[Task: C1 Begins execution                          at t = 7]
[Task: C1 Ends execution                            at t = 8]
[Task: C1 Calls server:                    S1       at t = 8]
[Task: S1 Ends execution on behalf of:     C5       at t = 9]
[Task: S1 Begins execution on behalf of:   C1       at t = 9]
[Task: S1 Ends execution on behalf of:     C1       at t = 10]
[Task: C1 Begins execution                          at t = 10]
[Task: C1 Ends execution                            at t = 11]
[Task: C2 Ends execution                            at t = 12]
[Task: S2 Begins execution on behalf of:   C3       at t = 12]
[Task: S2 Ends execution on behalf of:     C3       at t = 13]
[Task: C3 Begins execution                          at t = 13]
[Task: C3 Ends execution                            at t = 14]
[Task: C4 Ends execution                            at t = 15]
[Task: C5 Begins execution                          at t = 15]
[Task: C5 Ends execution                            at t = 16]
******************* Test Complete *******************
```

Figure B-26 Expected Events for PC_13

Appendix C Example of Test Specification

```
--++*****************************************************************
--.NAME.
--  tstchb.ada
--
--.TITLE.
--  TeST_CHaracteristics Body
--
--.DESCRIPTION.
--
--.VERSION.
--  4.2 modified on 2/28/91
--
--.COMMENTS.
--
--
--.EXTERNAL_PACKAGES.
--  PACKAGE NAME        PARAMETERS          USAGE
--  ============        ==========          =====
--  Generator_Structure
--  Customize
--
--.INTERNAL_PACKAGES.
--  PACKAGE NAME        PARAMETERS          USAGE
--  ============        ==========          =====
--
--
--++*****************************************************************
with Generator_Structure;
use Generator_Structure;
package body Test_Characteristics is
begin
  Test_Suite :=
      (Number_Of_Tests => 25,
       Test =>
 (1 => . . .
  . . .
  8 =>
  :   (Number_Of_Tasks  => 5,
  :   :Main_Unit_Name   => Set_String ("bi_05"),
  :   :Generated_File   => Set_String ("bi_05.ada"),
  :   :Scheduling       => BI,
  :   :Tasks => ((Task_Type          => Client,
  :    :             Task_Name          => C1,
  :    :             Offset_Start_Time  => 6,
  :    :             Priority           => P10,
  :    :             Execution_Pattern  =>
  :    :               (Number_Of_Events  => 3,
  :    :                Event_Array       =>
  :    :                  (1 => (Client_Execution, 1),
  :    :                   2 => (Server_Call, S1, E1),
  :    :                   3 => (Client_Execution, 1)
  :    :                  )
  :    :               )
  :    :             ),
(Task_Type            => Client,
 Task_Name            => C2,
 Offset_Start_Time    => 5,
 Priority             => P9,
 Execution_Pattern    =>
   (Number_Of_Events       => 1,
    Event_Array            =>
      (1 => (Client_Execution, 2)
      )
   )
),

(Task_Type            => Client,
 Task_Name            => C3,
 Offset_Start_Time    => 1,
 Priority             => P8,
 Execution_Pattern    =>
   (Number_Of_Events       => 3,
    Event_Array            =>
      (1 => (Client_Execution, 1),
       2 => (Server_Call, S1, E2),
       3 => (Client_Execution, 1)
      )
   )
),

(Task_Type            => Server,
 Task_Name            => S1,
 Entries              =>
   (Number_Of_Entries      => 2,
    Entry_Array            =>
      (1 => (Entry_Name         => E1,
             Execution_Pattern =>
               (Number_Of_Events => 1,
                Event_Array      =>
                  (1 => (Server_Execution, 1)
                  )
               )
            ),
       2 => (Entry_Name         => E2,
             Execution_Pattern =>
               (Number_Of_Events => 3,
                Event_Array      =>
                  (1 => (Server_Execution, 2),
                   2 => (Server_Call, S2, E1),
                   3 => (Server_Execution, 2)
                  )
               )
            )
      )
   )
),
   :    :               (Task_Type            => Server,
   :    :               :Task_Name            => S2,
   :    :               : Entries             =>
   :    :               :  (Number_Of_Entries   => 1,
   :    :               :  :Entry_Array         =>
   :    :               :  :   (1 => (Entry_Name      => E1,
   :    :               :  :    :     :Execution_Pattern =>
   :    :               :  :    :     :  (Number_Of_Events        => 1,
   :    :               :  :    :     :   Event_Array             =>
   :    :               :  :    :     :      (1 => (Server_Execution, 2)
   :    :               :  :    :     :      )
   :    :               :  :    :     :  )
   :    :               :  :    :     )
   :    :               :  :    )
   :    :               :  )
   :    :               )
   :    :             )
   :    ),
   9 => . . .
   . . .
   25 => . . .
  ) );
end Test_Characteristics;
```

Appendix D Code for Test Case BI_05

This appendix shows the code generated for test case BI_05. The code is generated in a single file. We have split this file into subsections to help the reader identify the various generated components. The specification that was sent to the generator is given in Appendix C on page 87. The diagram describing this test case is given in Figure A-9 on page 36.

D.1 BI_05_Test_Harness Package Specification

```
--++****************************
-- Test-case file
--   built by Generator tool
--++****************************
--
with System;
with Calendar;   use Calendar;
with Harness_Constants;
package bi_05_Test_Harness is
   procedure Start_Run(Start_Time: in TIME);
   task Test_Done is
      entry Client_C1_Done;
      entry Client_C2_Done;
      entry Client_C3_Done;
      entry Complete;
      pragma Priority (Harness_Constants.Test_Done_Priority);
   end Test_Done;
end bi_05_Test_Harness;
----
```

D.2 BI_05_Client_C1_Package Specification

```
with System;
with Calendar;      use Calendar;
with Harness_Constants;
package bi_05_Client_C1_Package is
   task C1_Task is
      entry START(Start_Time : in TIME);
      pragma Priority (Harness_Constants.P10);
   end C1_Task;
end bi_05_Client_C1_Package;
----
```

D.3 BI_05_Client_C2_Package Specification

```
with System;
with Calendar;      use Calendar;
with Harness_Constants;
package bi_05_Client_C2_Package is
   task C2_Task is
      entry START(Start_Time : in TIME);
      pragma Priority (Harness_Constants.P9);
   end C2_Task;
end bi_05_Client_C2_Package;
----
```

D.4 BI_05_Client_C3_Package Specification

```
with System;
with Calendar;        use Calendar;
with Harness_Constants;
package bi_05_Client_C3_Package is
  task C3_Task is
    entry START(Start_Time : in TIME);
    pragma Priority (Harness_Constants.P8);
  end C3_Task;
end bi_05_Client_C3_Package;
----
```

D.5 BI_05_Server_S1_Package Specification

```
with Calendar;
with Vendor_Specifics;
with Harness_Constants; use Harness_Constants;
package bi_05_Server_S1_Package is
  task S1_Task is
    entry Get_Id(Task_Id : out Vendor_Specifics.Task_Id);
    entry E1 (Task_Number : in Task_ID_Type);
    entry E2 (Task_Number : in Task_ID_Type);
  end S1_Task;
end bi_05_Server_S1_Package;
----
```

D.6 BI_05 Server_S2_Package Specification

```
with Calendar;
with Vendor_Specifics;
with Harness_Constants; use Harness_Constants;
package bi_05_Server_S2_Package is
  task S2_Task is
    entry Get_Id(Task_Id : out Vendor_Specifics.Task_Id);
    entry E1 (Task_Number : in Task_ID_Type);
  end S2_Task;
end bi_05_Server_S2_Package;
----
```

D.7 BI_05_Test_Harness Package Body

```
with System;
with IO_Pkgs;
with Calendar;
with Harness_Constants;
with Vendor_Specifics;
with bi_05_Client_C1_Package;
with bi_05_Client_C2_Package;
with bi_05_Client_C3_Package;
with bi_05_Server_S1_Package;
with bi_05_Server_S2_Package;
package body bi_05_Test_Harness is
  S1_Task_Id : Vendor_Specifics.Task_Id;
  S2_Task_Id : Vendor_Specifics.Task_Id;
  procedure Start_Run(Start_Time : in TIME) is
    procedure Set_Server_Priorities is
      use Calendar;
    begin
      ---------
      -- Set the priority for each server, and, if PCP, set
      -- the priority ceiling.
      ---------
      Vendor_Specifics.change_priority(
                  S1_Task_Id,
                  Harness_Constants.Server_Priority);
      Vendor_Specifics.change_priority(
                  S2_Task_Id,
                  Harness_Constants.Server_Priority);
    end Set_Server_Priorities;
  begin
    bi_05_Server_S1_Package.S1_Task.Get_Id( S1_Task_id );
    bi_05_Server_S2_Package.S2_Task.Get_Id( S2_Task_id );
    Set_Server_Priorities;
    bi_05_Client_C1_Package.C1_Task.START(Start_Time);
    bi_05_Client_C2_Package.C2_Task.START(Start_Time);
    bi_05_Client_C3_Package.C3_Task.START(Start_Time);
  exception
    when OTHERS =>
      IO_Pkgs.Txt_Io.Put_Line("*** Unexpected Exception in Start_Run ***");
  end Start_Run;
  task body Test_Done is
  begin
    ---------
    -- Wait for each client to finish
    ---------
    accept Client_C1_Done;
    accept Client_C2_Done;
    accept Client_C3_Done;
    ---------
    -- Synchronize with Main program (Test is Done)
    ---------
    accept Complete;
  exception
    when OTHERS =>
      IO_Pkgs.Txt_Io.Put_Line("*** Unexpected Exception in Test_Done Task ***");
  end Test_Done;
end bi_05_Test_Harness;
---
```

D.8 BI_05_Client_C1_Package Body

```
with bi_05_Test_Harness;
with Harness_Support;
with IO_Pkgs;
with Harness_Event_Log_Manager; use Harness_Event_Log_Manager;
with Harness_Constants;        use Harness_Constants;
with bi_05_Server_S1_Package;
package body bi_05_Client_C1_Package is
  task body C1_Task is
    Init           : constant DURATION :=    6.0;
    Exec_Time_1  : constant DURATION :=    1.0;
    Exec_Time_2  : constant DURATION :=    1.0;
    C1_Task_Start: TIME;
  begin
    --------
    -- Wait for Start_Time (Reference to time = 0) & then
    --  Suspend client until it is time to begin execution
    --------
    accept START(Start_Time : in TIME) do
      C1_Task_Start := Start_Time + Init;
    end START;
    Harness_Support.Suspend_Task(C1_Task_Start);
    --------
    -- Client execution
    --------
    Log_Event(C1. None, Client_Exec_Begins);
    Harness_Support.Spend_Time(Exec_Time_1);
    Log_Event(C1, None, Client_Exec_Ends);
    --------
    -- Call Server
    --------
    Log_Event(C1, S1, Server_Call_Begins);
    bi_05_Server_S1_Package.S1_Task.E1(C1);
    --------
    -- Client execution
    --------
    Log_Event(C1, None, Client_Exec_Begins);
    Harness_Support.Spend_Time(Exec_Time_2);
    Log_Event(C1, None, Client_Exec_Ends);
    bi_05_Test_Harness.Test_Done.Client_C1_Done;  -- Signal Client Finished
  exception
    when OTHERS =>
      IO_Pkgs.Txt_Io.Put_Line("*** Unexpected Exception in bi_05_Client_C1 Task ***");
  end C1_Task;
end bi_05_Client_C1_Package;
--
```

D.9 BI_05_Client_C2_Package Body

```
with bi_05_Test_Harness;
with Harness_Support;
with IO_Pkgs;
with Harness_Event_Log_Manager; use Harness_Event_Log_Manager;
with Harness_Constants;        use Harness_Constants;
package body bi_05_Client_C2_Package is
  task body C2_Task is
    Init             : constant DURATION :=   5.0;
    Exec_Time_1  : constant DURATION :=   2.0;
    C2_Task_Start: TIME;
  begin
    --------
    -- Wait for Start_Time (Reference to time = 0) & then
    --  Suspen.` client until it is time to begin execution
    --------
    accept START(Start_Time : in TIME) do
      C2_Task_Start := Start_Time + Init;
    end START;
    Harness_Support.Suspend_Task(C2_Task_Start);
    --------
    -- Client execution
    --------
    Log_Event(C2, None, Client_Exec_Begins);
    Harness_Support.Spend_Time(Exec_Time_1);
    Log_Event(C2, None, Client_Exec_Ends);
    bi_05_Test_Harness.Test_Done.Client_C2_Done;  -- Signal Client Finished
  exception
    when OTHERS =>
      IO_Pkgs.Txt_Io.Put_Line("*** Unexpected Exception in bi_05_Client_C2 Task ***");
  end C2_Task;
end bi_05_Client_C2_Package;
----
```

D.10 BI_05_Client_C3_Package Body

```
with bi_05_Test_Harness;
with Harness_Support;
with IO_Pkgs;
with Harness_Event_Log_Manager; use Harness_Event_Log_Manager;
with Harness_Constants;       use Harness_Constants;
with bi_05_Server_S1_Package;
package body bi_05_Client_C3_Package is
  task body C3_Task is
    Init          : constant DURATION :=   1.0;
    Exec_Time_1   : constant DURATION :=   1.0;
    Exec_Time_2   : constant DURATION :=   1.0;
    C3_Task_Start: TIME;
  begin
    --------
    -- Wait for Start_Time (Reference to time = 0) & then
    --  Suspend client until it is time to begin execution
    --------
    accept START(Start_Time : in TIME) do
      C3_Task_Start := Start_Time + Init;
    end START;
    Harness_Support.Suspend_Task(C3_Task_Start);
    --------
    -- Client execution
    --------
    Log_Event(C3, None, Client_Exec_Begins);
    Harness_Support.Spend_Time(Exec_Time_1);
    Log_Event(C3, None, Client_Exec_Ends);
    --------
    -- Call Server
    --------
    Log_Event(C3, S1, Server_Call_Begins);
    bi_05_Server_S1_Package.S1_Task.E2(C3);
    --------
    -- Client execution
    --------
    Log_Event(C3, None, Client_Exec_Begins);
    Harness_Support.Spend_Time(Exec_Time_2);
    Log_Event(C3, None, Client_Exec_Ends);
    bi_05_Test_Harness.Test_Done.Client_C3_Done;  -- Signal Client Finished
  exception
    when OTHERS =>
      IO_Pkgs.Txt_Io.Put_Line("*** Unexpected Exception in bi_05_Client_C3 Task ***");
  end C3_Task;
end bi_05_Client_C3_Package;
---
```

D.11 BI_05_Server_S1_Package Body

```
with Harness_Support;
with Calendar;              use Calendar;
with Harness_Event_Log_Manager; use Harness_Event_Log_Manager;
with Harness_Constants;     use Harness_Constants;
with bi_05_Server_S2_Package;
package body bi_05_Server_S1_Package is
  task body S1_Task is
    Exec_Time_1: constant DURATION :=  1.0;
    Exec_Time_2: constant DURATION :=  2.0;
    Exec_Time_3: constant DURATION :=  2.0;
  begin
    ---------
    -- Get my id
    ---------
    accept Get_Id (Task_Id : out Vendor_Specifics.Task_Id) do
      Task_Id := Vendor_Specifics.Get_Task;
    end Get_Id;
    ------
    -- Endless loop servicing clients
    ------
    loop
      select
        accept E1 (Task_Number : in Task_ID_Type) do
          ----------
          -- Server Execution
          ----------
          Log_Event(S1, Task_Number, Server_Exec_Begins);
          Harness_Support.Spend_Time(Exec_Time_1);
          Log_Event(S1, Task_Number, Server_Exec_Ends);
        end E1;
      or
        accept E2 (Task_Number : in Task_ID_Type) do
          ----------
          -- Server Execution
          ----------
          Log_Event(S1, Task_Number, Server_Exec_Begins);
          Harness_Support.Spend_Time(Exec_Time_2);
          Log_Event(S1, Task_Number, Server_Exec_Ends);
          ----------
          -- Server Call
          ----------
          Log_Event(S1, S2, Server_Call_Begins);
          bi_05_Server_S2_Package.S2_Task.E1(S1);
          ----------
          -- Server Execution
          ----------
          Log_Event(S1, Task_Number, Server_Exec_Begins);
          Harness_Support.Spend_Time(Exec_Time_3);
          Log_Event(S1, Task_Number, Server_Exec_Ends);
        end E2;
      or
        terminate;    -- when server is no-longer Callable
      end select;
    end loop;
  end S1_Task;
end bi_05_Server_S1_Package;
---
```

D.12 BI_05_Server_S2_Package Body

```
with Harness_Support;
with Calendar;              use Calendar;
with Harness_Event_Log_Manager; use Harness_Event_Log_Manager;
with Harness_Constants;     use Harness_Constants;
package body bi_05_Server_S2_Package is
  task body S2_Task is
    Exec_Time_1: constant DURATION :=   2.0;
  begin
    ------
    -- Get my id
    ------
    accept Get_Id (Task_Id : out Vendor_Specifics.Task_Id) do
      Task_Id := Vendor_Specifics.Get_Task;
    end Get_Id;
    ------
    -- Endless loop servicing clients
    ------
    loop
      select
        accept E1 (Task_Number : in Task_ID_Type) do
          ----------
          -- Server Execution
          ----------
          Log_Event(S2, Task_Number, Server_Exec_Begins);
          Harness_Support.Spend_Time(Exec_Time_1);
          Log_Event(S2, Task_Number, Server_Exec_Ends);
        end E1;
      or
        terminate;   -- when server is no-longer Callable
      end select;
    end loop;
  end S2_Task;
end bi_05_Server_S2_Package;
----
```

D.13 Main Procedure for BI_05

```
with bi_05_Test_Harness;
with Harness_Event_Log_Manager;
with Harness_Constants;
with Harness_Support;
with Calendar;              use Calendar;
procedure bi_05 is
  pragma Priority (Harness_Constants.Main_Priority);
  Start_Time : TIME;
begin
  Harness_Support.Calibrate_Spend_Time;                -- Calibration
  Harness_Event_Log_Manager.Initialize(Start_Time);    -- Initialize Logger
  bi_05_Test_Harness.Start_Run(Start_Time);            -- Start Clients
  bi_05_Test_Harness.Test_Done.Complete;               -- Wait for Clients
  Harness_Event_Log_Manager.Print_Time_Lines;          -- Print Time-line
  Harness_Event_Log_Manager.Quit;                      -- Stop Logger
end bi_05;
```

Appendix E Obtaining Source Code and Information

Contact Keith Kohout at Naval Weapons Center Code 3916 (619-939-1278; e-mail: keith@sol.nwc.navy.mil) for additional information and access to the software. The tests are also available by anonymous FTP from the SEI (ftp.sei.cmu.edu) in the directory pub/RMA-Validation-Tests.

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S CMU/SEI-92-TR-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-92-TR-1
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (City, State and ZIP Code) ESD/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESD/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO 63756E; PROJECT NO. N/A; TASK NO N/A; WORK UNIT NO. N/A

11. TITLE (Include Security Classification)
Ada Validation Tests for Rate Monotonic Scheduling Algorithms

12. PERSONAL AUTHOR(S)
Keith A. Kohout, Kent Meyer, John B. Goodenough

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM TO	February 1992	107

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)

Ada	Rate Monotonic
basic inheritance	real-time systems
priority ceiling	task scheduling

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report presents a set of tests for checking whether an Ada runtime system properly supports certain rate monotonic scheduling algorithms, specifically, the basic inheritance and priority ceiling protocols. These tests are intended to be used by vendors and by users to validate implementations of these protocols. The report describes the tests and how they are to be used. The source code is available electronically.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL John S. Herman, Capt, USAF	22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631 22c. OFFICE SYMBOL ESD/AVS (SEI)

DD FORM 1473, 83 APR EDITION of 1 JAN 73 IS OBSOLETE UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS

ABSTRACT —continued from page one, block 19